HOBO Wireless Sensor Network

Evan McKeon

FGCU

Computer Networks

Instructor: Dr. J. Zalewski

November 20, 2011

1. Introduction

The HOBO project is a continuation of a previous project, worked on by Lavanda[1]. The device used for this project is the Onset HOBO micro weather station. A data logger, temperature sensor, relative humidity/pressure sensor, and two RF radio modules are included in the HOBO device. One radio module and all sensors connect to the data logger, while the other radio module connects to a server. A set up of the devices is shown in Figure 1 below[1].

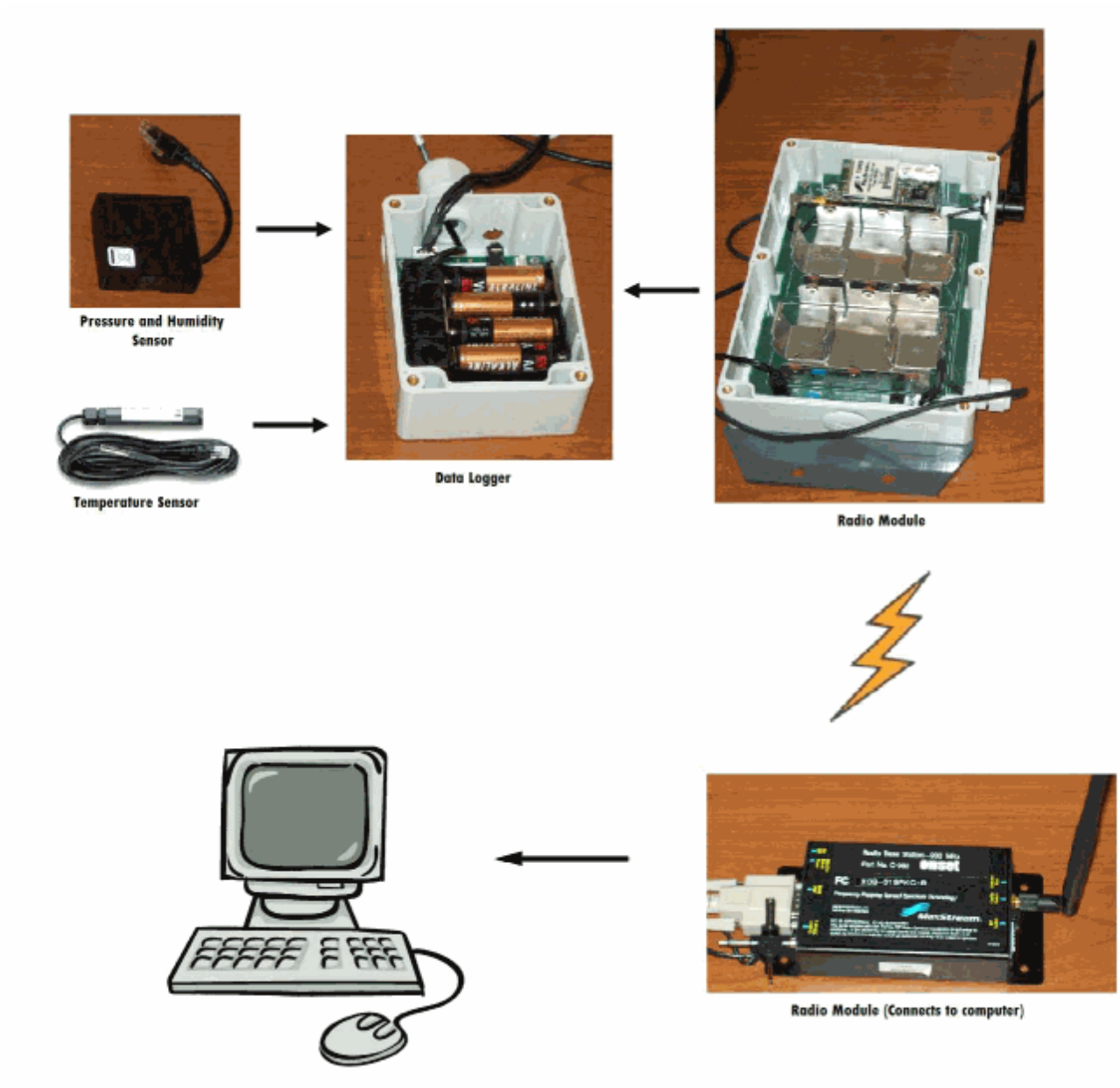


Figure 1 - Device Setup

The device also comes with two programs that must be installed. One is the Onset Remote Site Manager, which sets tasks for the device and deals with communicating back to the server. The other is Onset's HOBOware software, which is used to set up the data logger and the attached sensors. HOBOware also provides powerful graphing features let allow for plotting and filtering data[2]. An example of the HOBOware graphing functions is shown below in Figure 2[3].The previous project author also custom built a Java application for automatically updating a database with the data from the device. This application is named HOBOBuilder and because the data is gathered conveniently through this program, it is then used to update a website. The website shows all the recent findings of the HOBO device in an organized way, in order for a user to find information relatively easily.

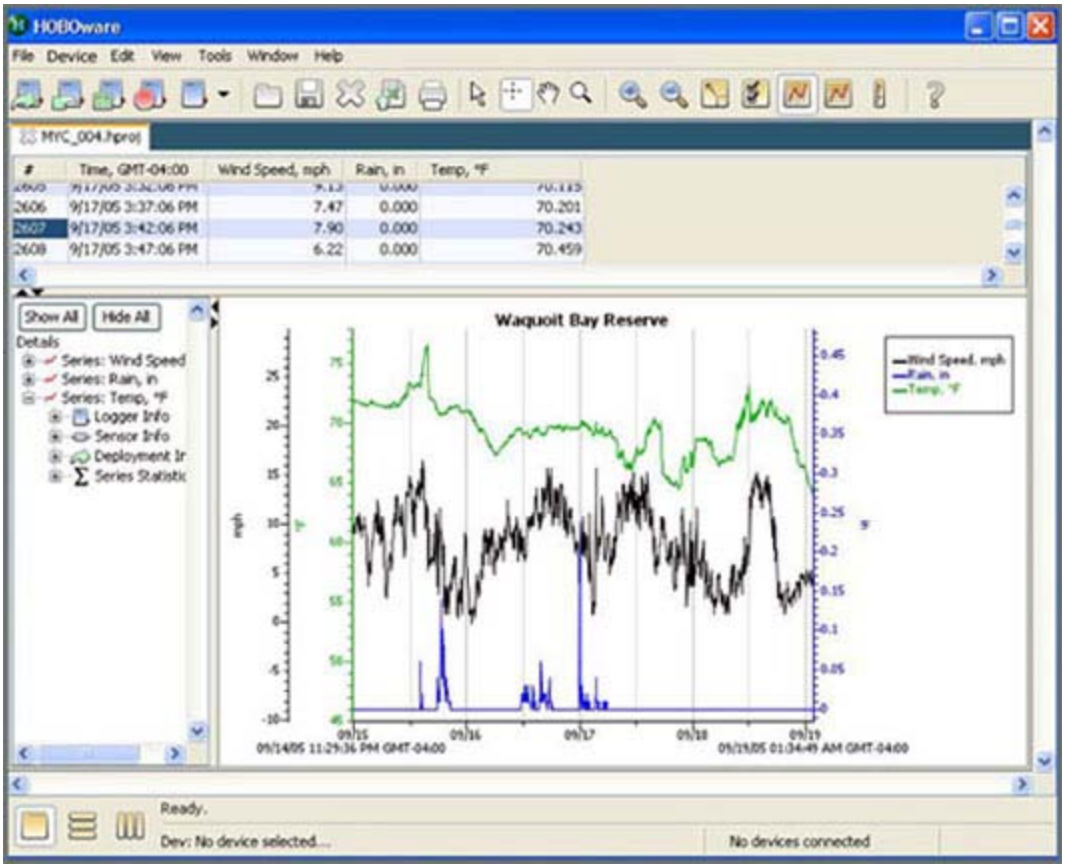


Figure 2 - Example Graph

2. Definition of the Problem

2.1 Problem Introduction

The HOBO monitoring station requires the use of the two software provided, HOBOware and the Onset remote monitoring program. HOBOware connects the logger, sets up the recording schedule and outputs data. The remote monitoring program is required for connecting the device wirelessly, setting the recording schedule and helps to read out data. The HOBO software allows for output to a text file, XML file or HOBO-specific file.

2.2 Previous Problems

Because of the requirements for being able to use the device, the previous projects on this device dealt mostly with accessibility. As the device only outputs onto the computer it is set up on by default, if one wishes to use it, one must access that specific computer. Since the device is a wireless system designed to make the readings easier to get to due to the possible remote placement of the device, this set up was not sufficient. Therefor, the previous project focused on two goals. The first goal was creating a website to allow access to the HOBO readings that could be filtered to any user with a browser. The second goal was to create a program that can access and read the HOBO output, which allows the website to retrieve this information more easily. The final logical configuration after the previous project can be seen in Figure 3[1].

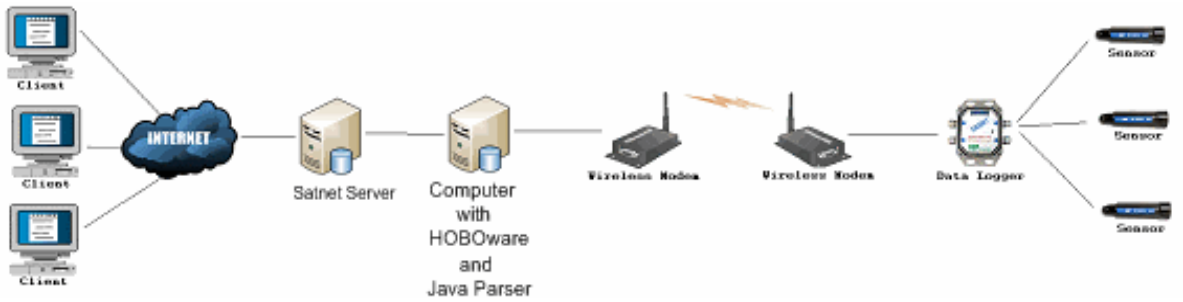


Figure 3 - Logical Diagram

2.3 Current Problem

A lot has been accomplished in the previous iterations of the project. Part of this project is updating and improving upon the work that has already been done. The HOBO device can also allow for different measurements than just the ones used in the previous project, however new sensors would

have to be bought. The current sensors are for Temperature, Barometric Pressure and Relative Humidity. The data logger also supports the following measurements that are not in the lab: 4-20mA, DC Voltage, Kilowatt Hours (kWh), Leaf Wetness, Light Intensity, Pulse Input, Rainfall, Soil Moisture, Water Flow, Wind. However, after researching the extra sensors the price for those not already in use is very high, so there are no other sensors on this iteration of the project.

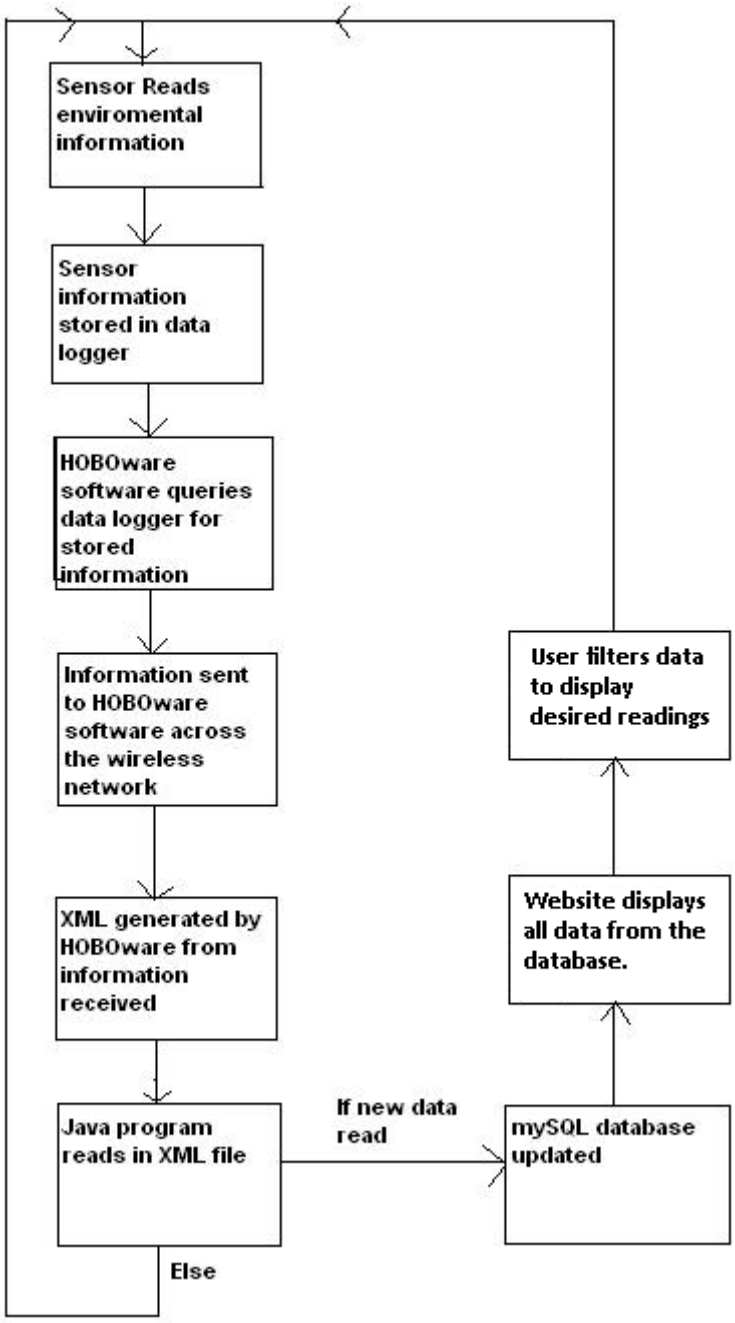
Some of the work that had been done in the previous semesters needs to be redone due to a new computer. None of the work has been lost, but the new computer must be set up with the HOBO programs as well as the previous project's application in order for not only the device, but for the website to be working properly. The two different HOBO software elements (HOBOware and Onset remote site manager) must be installed and then setup with the device. The database must be reconnected to the website and the software by putting the previously created Java software (HoboBuilder) on the computer as well.

This project can make use of new measurements and find a way to integrate the results into the existing website and perhaps provide some sort of analysis, such as in the form of graphs and charts. If another custom application would help the functionality or provide some benefit, that is also something that can be done, in similar fashion to the HOBOBuilder already created. As much has been done already with the device, the direction for this project is to continue to extend the functionality of the device. In other words, provide even more accessibility to users and provide something more attractive to interact with. This will hopefully broaden the range of possible users and create an interesting feature to garner the attention of such users. The current website works on computers, but is not set up to be accessed and used from mobile devices. Rather than recreate and/or update the website which serves its purpose well, there will be a mobile website created in order to allow users to check on HOBO data from anywhere. The mobile site will allow users to get current information from the HOBO device as well as provide a link to the full site developed in the previous project.

3. Prospective Solution

3.1 Previous Solutions

The previous projects centered around building a website to better represent the HOBO device data and to make it more accessible, and then extending that website. A Java program named HoboBuilder was made in order to read the data output by the device and provide a way for the website to use this data. The HoboBuilder program takes the data in a txt file saved by the HOBO software and saves it to a mySQL database. The website then pulls information from that database to populate the charts and graphs. This process is shown in Figure 3[1].


Sensor Reads enviromental information
Sensor information stored in data logger
HOBOware software queries data logger for stored information
Information sent to HOBOware software across the wireless network
XML generated by HOBOware from information received
Java program reads in XML file
If new data read
mySQL database updated
Website displays all data from the database.
User filters data to display desired readings
Else

3.2 Current Solution

First off, the software will be installed and setup on the new computer. Then the database must be made sure to be updating with the device data properly and be able to be referenced by the Java programs and website. The way to set up the device can be seen in the user manual. In order to solve the other issues, as mentioned before, a mobile website will be created. This website will provide the flexibility of being used on a phone as well as being accessed on a computer. The website will feature the data for each sensor of the HOBO device as well as provide a graphical display of the current amount. Design will be an important part of the website as it should be both simple and eye-catching. Mobile websites need to look good, but also have limited space to do it in. Of course, this will all have to be done once the database is up and running correctly.

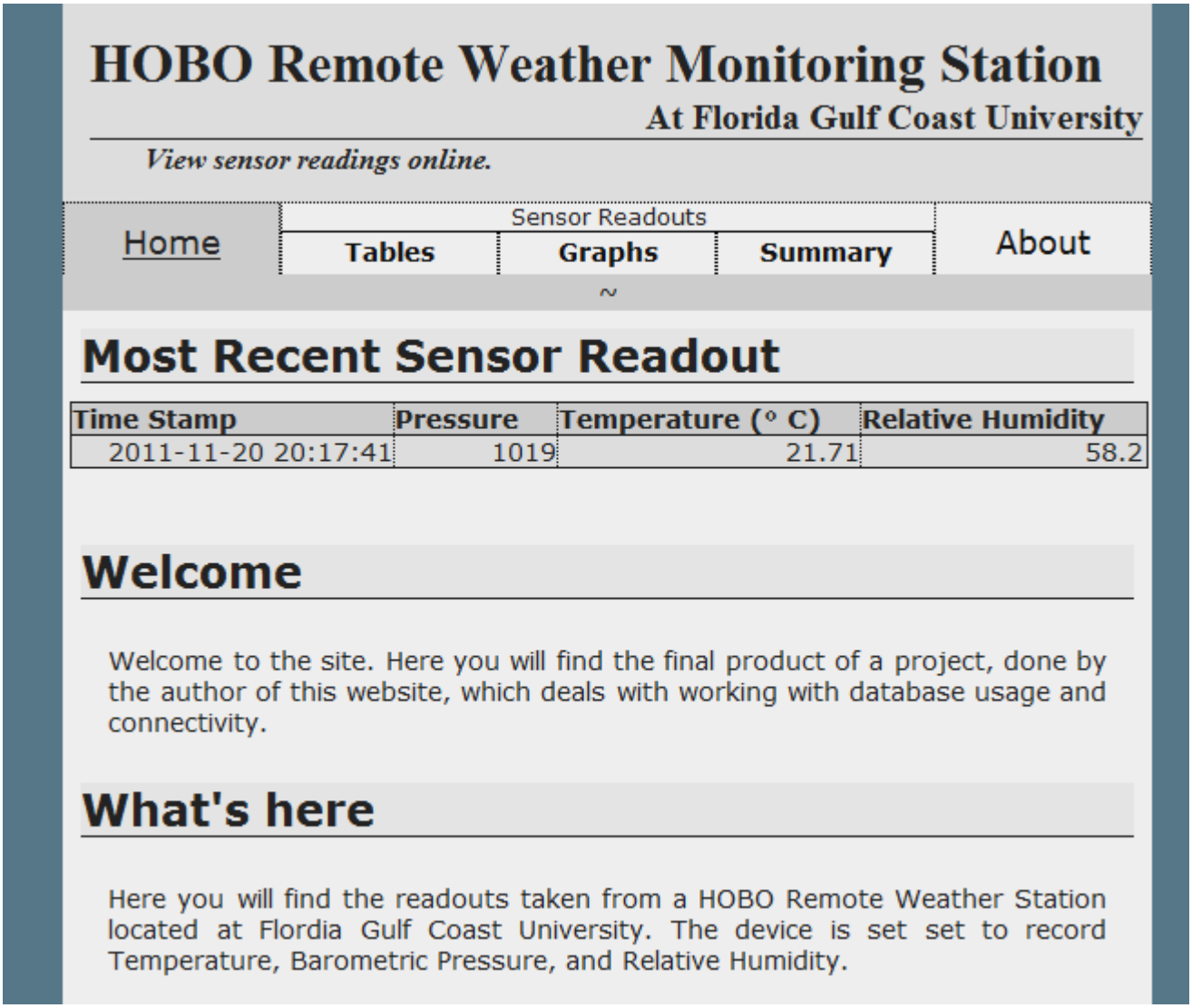

HOBO Remote Weather Monitoring Station
At Florida Gulf Coast University
View sensor readings online.
Home
Sensor Readouts
Tables
Graphs
Summary
About
Most Recent Sensor Readout
Time Stamp
Pressure
Temperature (° C)
Relative Humidity
2011-11-20 20:17:41
1019
21.71
58.2
Welcome
Welcome to the site. Here you will find the final product of a project, done by the author of this website, which deals with working with database usage and connectivity.
What's here
Here you will find the readouts taken from a HOBO Remote Weather Station located at Flordia Gulf Coast University. The device is set set to record Temperature, Barometric Pressure, and Relative Humidity.

Figure 4 - Portion of Current Website

Although recreating the current website was considered, that idea proved to be unnecessary of pursuing. The previous project website (Figure 4) does what it intends to well. The website shows off what the device can do and accurately represents the HOBO project. That being said, the way to move forward with this project is to provide something that could be even more useful to a larger base of users. Also the previous goal of making something more eye-catching remains intact. A new website built specifically for mobile devices fits these goals perfectly.

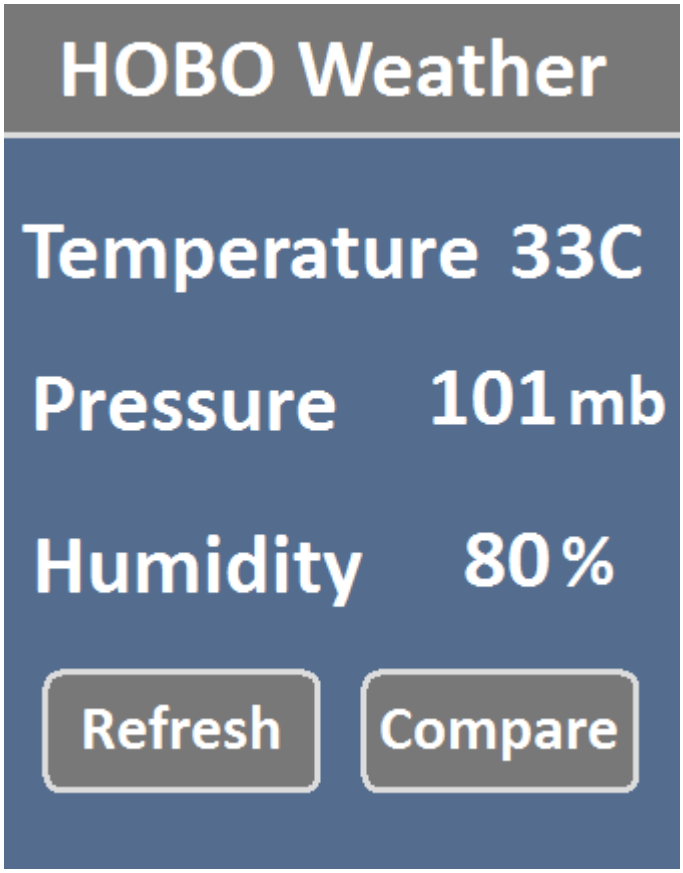


Figure 5 - Mobile Website Prototype

More and more people are starting to use phones as a primary way to access the internet. Companies are now focusing much more on phone capable software and investing much more into mobile software development. Along with a mobile website being current and cutting edge it provides ease of use, especially for a device like the HOBO where just getting the readings is of primary concern. The HOBO is basically a weather device and for users to be able to access up to the minute weather information is actually something of value. The mobile website will provide the flexibility for users to check data from anywhere, while still being able to go online to the larger website to access

more in depth analysis of the HOBO numbers.

The concept for the mobile website is to design a simple, easy to use, phone capable site for users to be able to access quickly. The prototype drawn up for a site that would meet those specifications can be seen in Figure 4.

4. Implementation

The prospective solution will be implemented by using a combination of PHP, CSS, MySQL, and HTML. The site will connect to the previously established database in order to pull the values from the HOBO device. The mobile website contains five PHP pages in all: the index, one page for each sensor, and an about page.

4.1 Index

The index contains the title, a button for each sensor, and a button for the about page (Figure 5). The index is the home page for the site and keeps it simple as it is the first page the user will view.

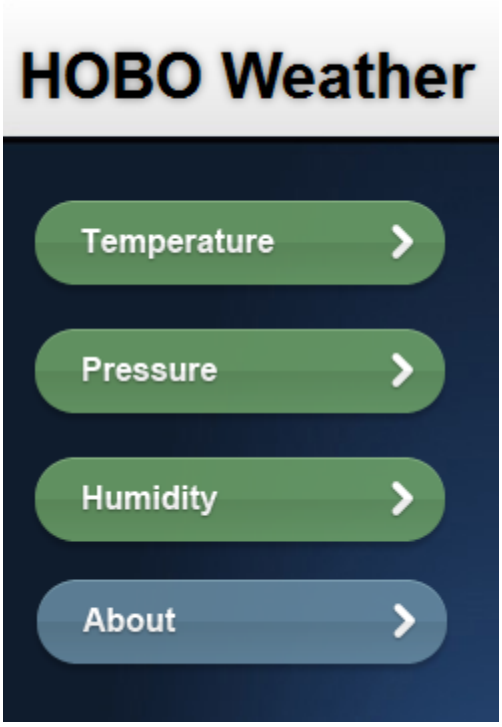


Figure 6 - Index

From the index there are four possible actions (selections) for the user which coincide with the four buttons:

- Temperature
- Pressure
- Humidity
- About

Selecting any of these buttons take the user to a page for specifically for that topic.

4.2 Temperature

Once the temperature button is selected the website will go to the temperature page (Figure 6) which shows a graphical display for temperature similar to a thermometer and also lists the temperature. From this page the only option for the user is to return to the previous page by clicking the green arrow. For the graph, the most recent output stored in the database is retrieved and is used as the amount to show.

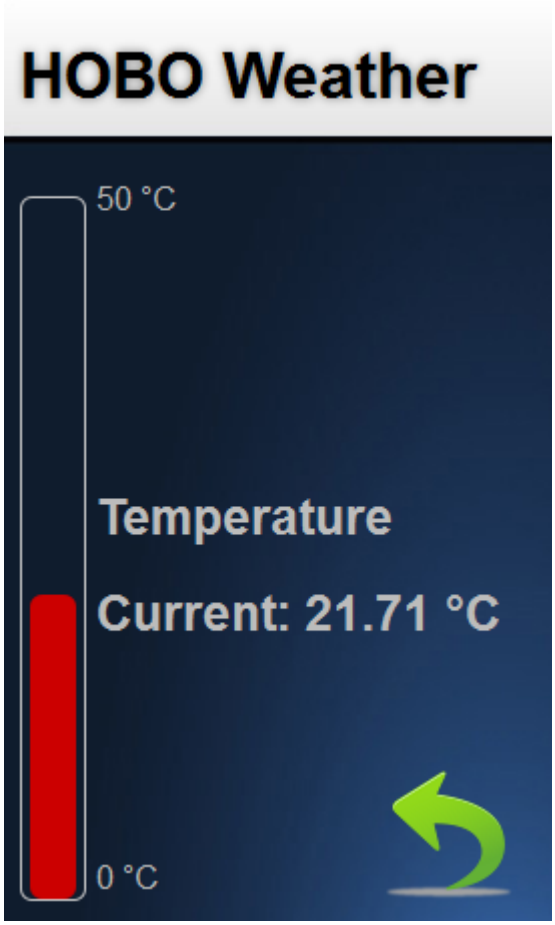


Figure 7 - Temperature

The code for the humidity page can be found in appendix I.

4.3 Pressure

Once the pressure button is selected the website will go to the pressure page (Figure 7) which shows a graphical display for pressure, much the same as the temperature page. Once again, the user's only option from here is to return to the index using the green arrow.

Figure 8 - Pressure

The code for the humidity page can be found in appendix J.

4.4 Humidity

The humidity page (Figure 7) is much like the previous two, and here the user has again the one option of going back by using the green arrow.

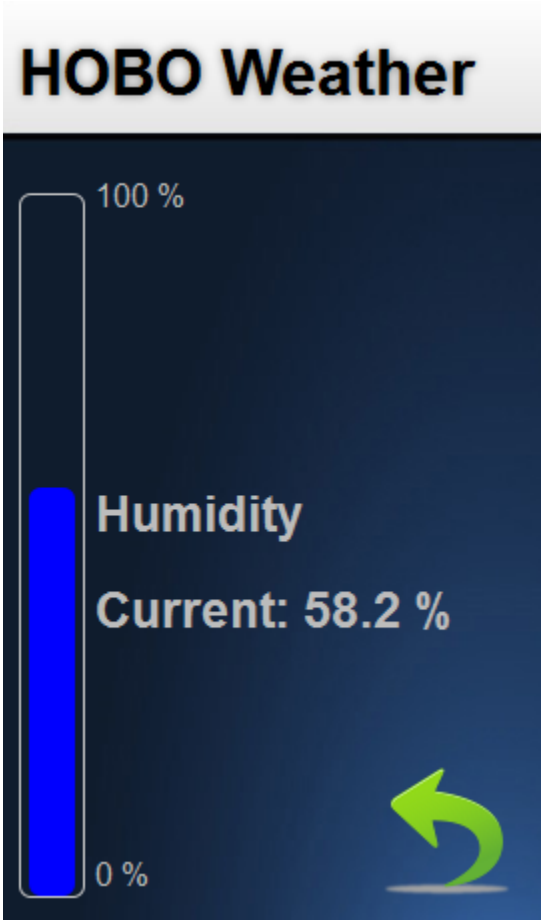


Figure 9 - Humidity

The code for the humidity page can be found in appendix K.

4.5 About

When the about button is selected, the website will take the user to a page displaying the author of the site as well as providing a link to the previous project's site. If a user is accessing from a computer and would prefer to see the whole capability of the previously designed site that person just needs to select the link shown. The user can also choose to return to the index of the site by the green arrow, as on the other pages.

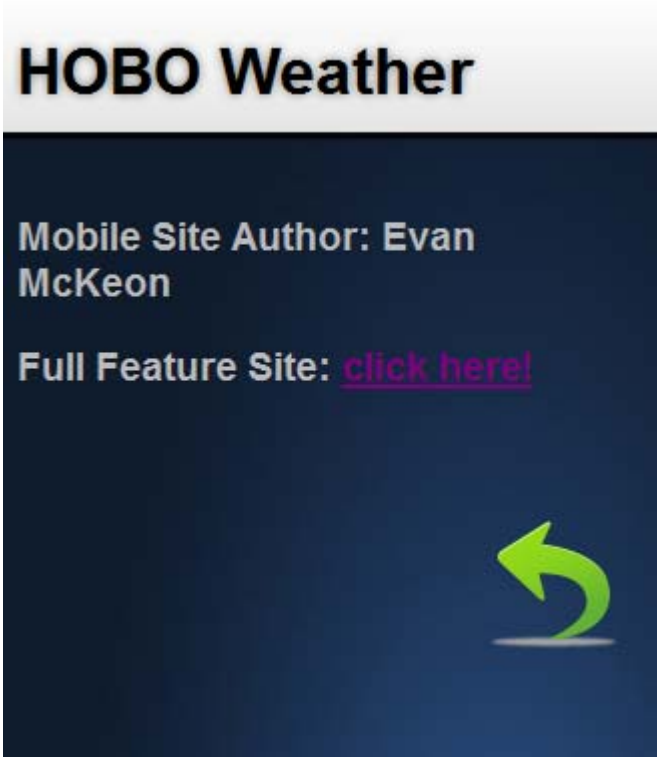


Figure 10 - About

This is the code for the about page:

```
<html>
<head>
<title>HOBO Mobile View</title>
<link href="mobile.css" rel="stylesheet" type="text/css"/>
<meta name="viewport" content="width=320" />
<body>
<div id="header">
<h1>HOBO Weather</h1>
</div>
<div>
<h3>Mobile Site Author: Evan McKeon</h3>
<h3>Full Feature Site: <a href="http://satnet.fgcu.edu/~nmlavand/projects/hobo/index.php">click here!</a></h3>
<div id="backArrow"><a href="index.php"><img src="back.png" /></a></div>
</div>
</body>
</html>
```

5. Conclusion

In conclusion the goal of this project is to make the data of the HOBO weather station more accessible and interesting to a wider user base. The mobile site will be uploaded to the satnet server where users can view it. The project makes the device and the established system, including the currently updated system from the previous project even more easily reached by clients. Anyone can now access the current weather report from the device from any internet capable mobile device. Now not only is the device wireless, but the wireless capabilities of the station are even greater.

My experience with the project and device has been frustrating at times, but also encouraging at times. I am very thankful for the previous work that has been done, because the Java program to grab the data from the HOBO device and add to the MySQL database is extremely helpful. However, because this project was so far along the road and had accomplished so much, it was very difficult to figure out which direction to take it in. Ultimately, I think I came up with and made something worthwhile and beneficial, but it was hard to get to that point. I had little knowledge of PHP and slightly more of MySQL going into the project, so that made it challenging at first. What made it especially challenging is I thought I would take it in a direction with another Java program at first, but that turned out not to be feasible, so I had to become more familiar with these languages in less time than I thought I was going to have initially. I think maybe the only way left to expand upon this project may be to add more variety of sensors, however I think while that may add usefulness to the device, it probably would not prove challenging enough for an entire project.

6. References

1. Lavanda, User Manual for HOBO Project, FGCU, April 15, 2010, <http://itech.fgcu.edu/faculty/zalewski/CNT4104/Projects/HOBO_User_Manual-Apr2010.pdf>

2. Onset, HOBO Micro Station Data Logger - H21-002, Onset Computer Corporation , <http://www.onsetcomp.com/products/data-loggers/h21-002>

3. Dalotech, HOBOware Software, <http://www.dalotech.com/hobo/software/hoboware.php>

7. User Manual

1. Hardware Configuration

Working with the Onset HOBO micro weather station, this project's objective is to use the sensors to detect weather conditions, record this information in a database, and make it available via the web . The Onset HOBO micro weather station consists of a data logger, a temperature sensor, a relative humidity/pressure sensor, and two RF radio modules. Figure 1 shows the different pieces of the HOBO micro weather station.

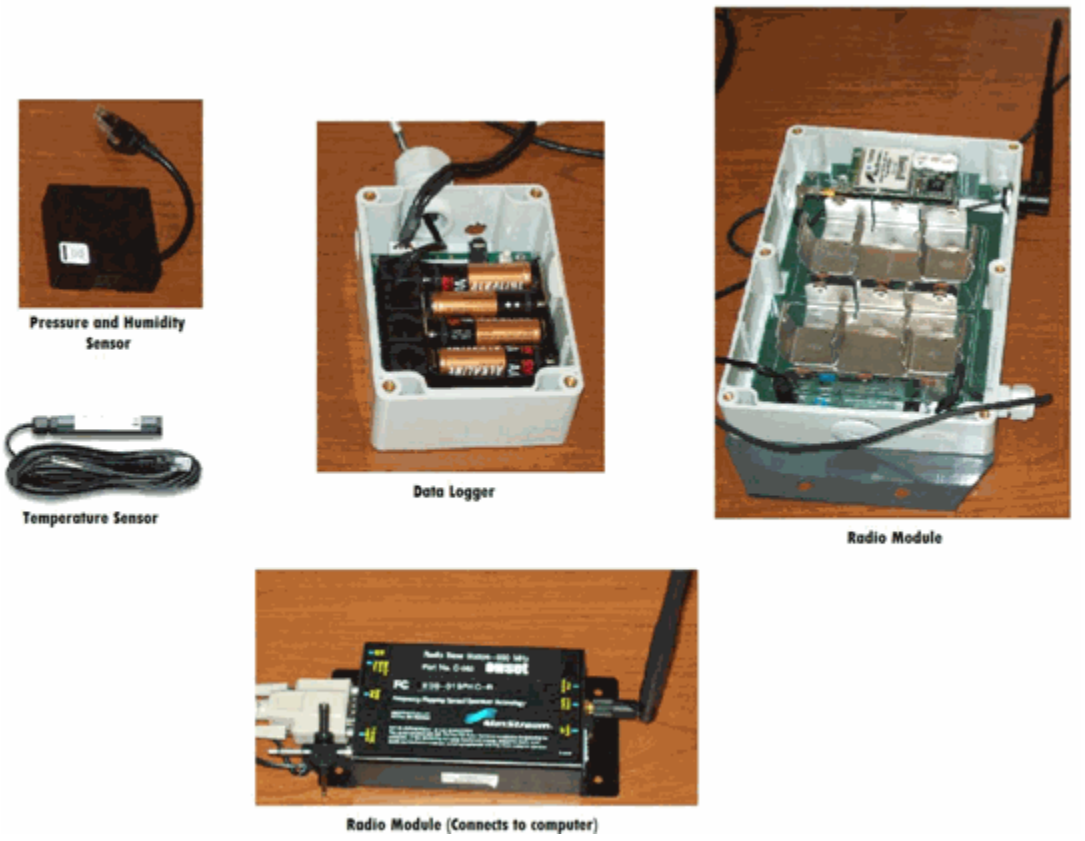


Figure 1 – Onset HOBO micro weather station devices

The sensors connect directly to the data logger. Four sensors can be plugged into the data logger at one time. They use a connection much like an Ethernet plug and plug into the data logger next to the batteries. The remote RF radio module connects to the data logger through a custom connection. The other RF radio module connects to a server through a serial port (RS 232).

2. HOBO Software Installation

In order to talk to the weather station one has to use Onset's proprietary software. First install the Onset Remote Site Manager. To install this program insert the installation CD and run the setup.exe program, if it does not auto run. There is no advanced configuration needed for this program, so just remember where you install it to so you can run the program later.

The next program that needs to be installed is Onset's HOBOware software. To install this software insert the HOBOware CD and run the setup.exe program, if it does not auto run. HOBOware has no advanced configuration needed; just remember where it is installed to so it can be run at a later time.

3. HOBO Software Configuration: HOBOWare

There are two choices for communication with the HOBO: HOBOware and Remote Site Manager. For this project we will use both HOBOware and Remote Site Manager. First we need to setup the data logger to record information from the sensors. In order to do this we will use HOBOware. Start HOBOware by double clicking on the HOBOware icon. As a result, a screen shown in Figure 3 appears.

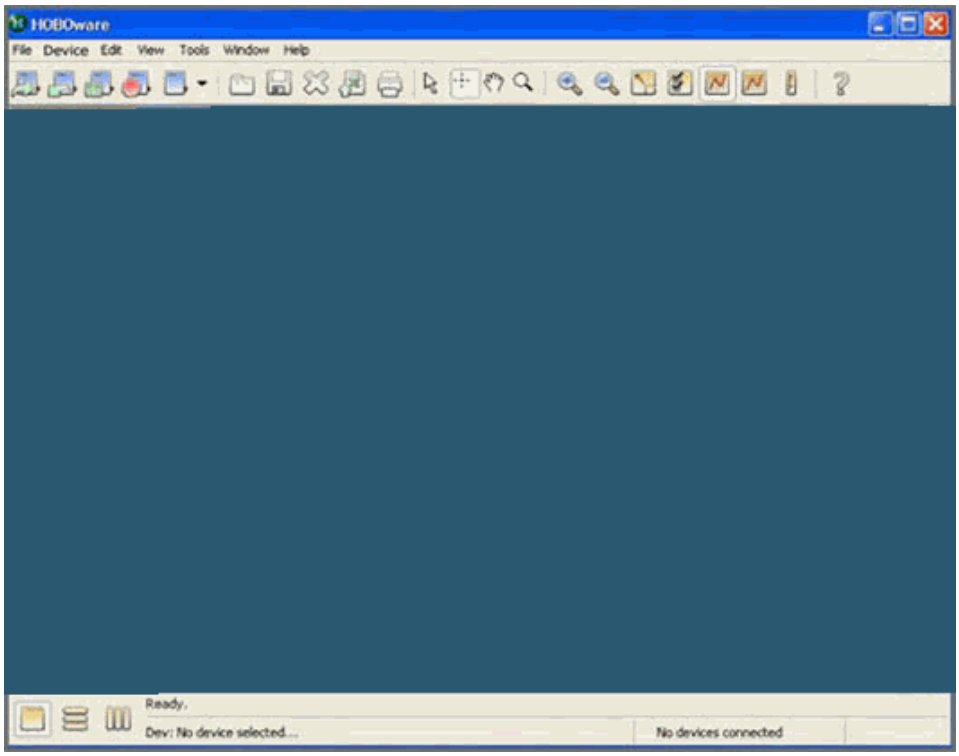


Figure 3 – HOBOware starting screen

Now at the top menu from Figure 3, go to device and select “Launch”. This will Launch Logger screen shown in Figure 4.

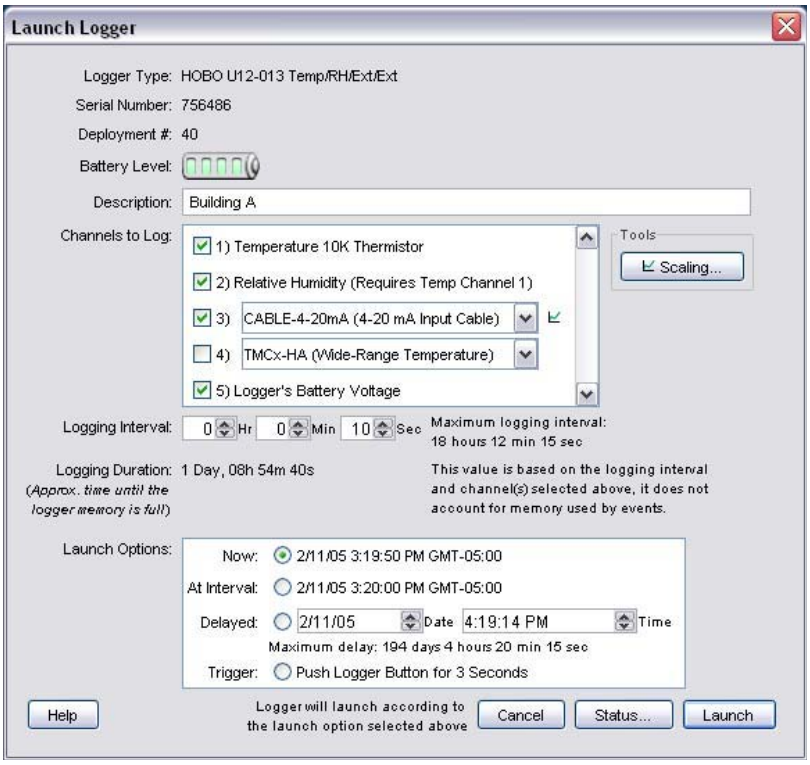

Launch Logger
Logger Type: HOBO U12-013 Temp/RH/Ext/Ext
Serial Number: 756486
Deployment #: 40
Battery Level:
Description: Building A
Channels to Log:
1) Temperature 10K Thermistor
2) Relative Humidity (Requires Temp Channel 1)
3) CABLE-4-20mA (4-20 mA Input Cable)
4) TMCx-HA (Wide-Range Temperature)
5) Logger's Battery Voltage
Tools
Scaling...
Logging Interval: 0 Hr 0 Min 10 Sec
Maximum logging interval: 18 hours 12 min 15 sec
Logging Duration: 1 Day, 08h 54m 40s
(Approx. time until the logger memory is full)
This value is based on the logging interval and channel(s) selected above, it does not account for memory used by events.
Launch Options:
Now: 2/11/05 3:19:50 PM GMT-05:00
At Interval: 2/11/05 3:20:00 PM GMT-05:00
Delayed: 2/11/05 Date 4:19:14 PM Time
Maximum delay: 194 days 4 hours 20 min 15 sec
Trigger: Push Logger Button for 3 Seconds
Help
Logger will launch according to the launch option selected above
Cancel
Status...
Launch

Figure 4 – Launch Logger screen from HOBOware

On this screen under “Channels to Log” make sure all of the desired sensors that you want to log are checked. It is recommended to select all of the sensors. Now select how often you would like the data logger to log sensor information by changing the “Logging Interval”. For this project we select 30 seconds. Once the Logging interval is selected we need to decide on a Launch option. Select the radial next to “Now” to start logging information immediately. Once you have done this, click on the “Launch” button at the bottom of the screen. Now the program will take some time to configure the data logger. Once the data logger is launched, which is indicated by the appearance of a status bar, you may close HOBOware, by selecting Exit menu choice from the screen as shown in Figure 3.

4. HOBO Software Configuration: Onset Remote Site Manager

Now start up the Onset Remote Site Manager, by double clicking on the Remote Site Manager icon. As a result a screen shown in Figure 5 should appear.

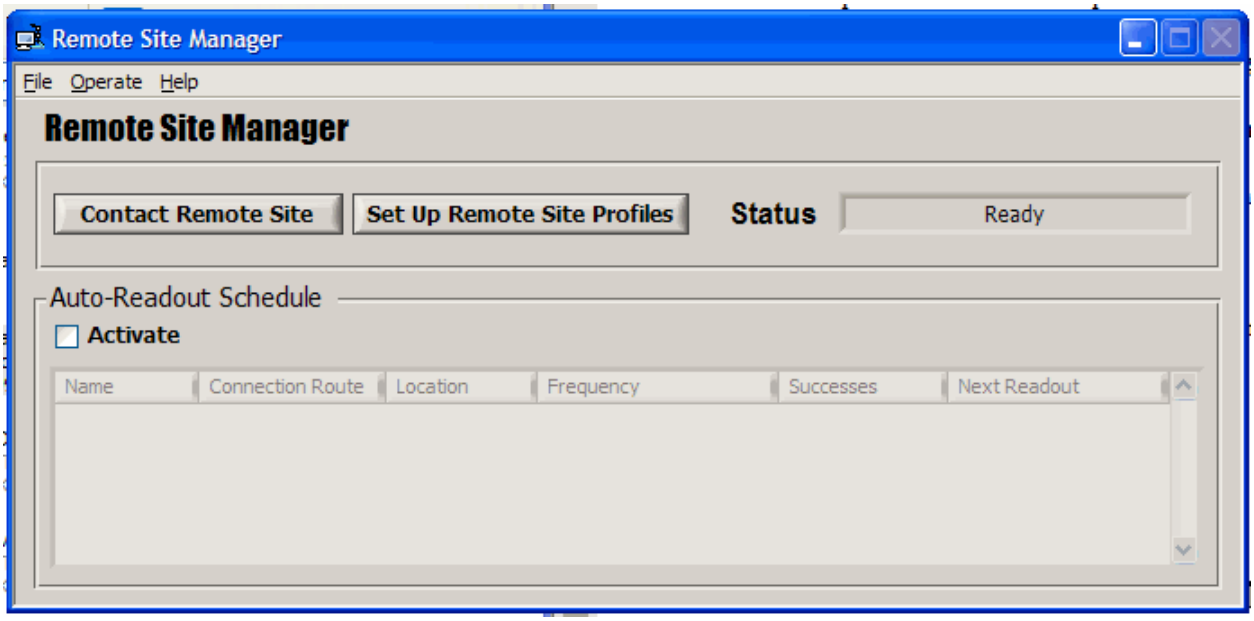


Figure 5 – Onset Remote Site Manager Start-up screen

Once the Remote Site Manager is running we need to create a remote site. In order to do this click on the “Set up remote site profiles” button and a screen will appear, as shown in Figure 6.

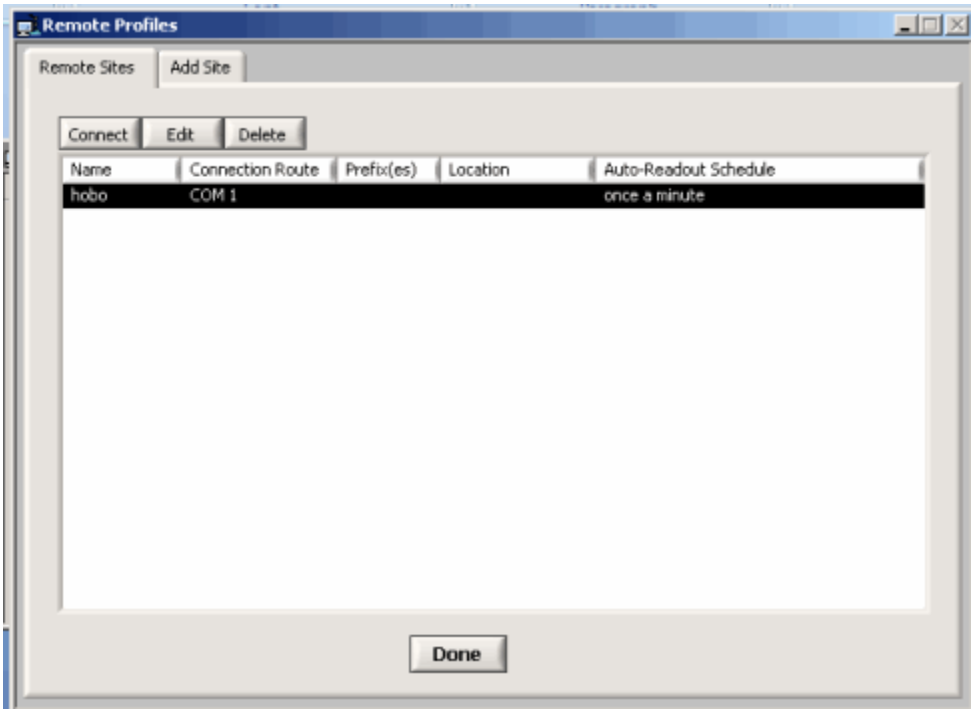


Figure 6 – Remote Profile page from Remote Site Manager program

From the Remote Profile Screen, as shown in Figure 6, select the “Add Site” tab. A screen like Figure 7 will appear.

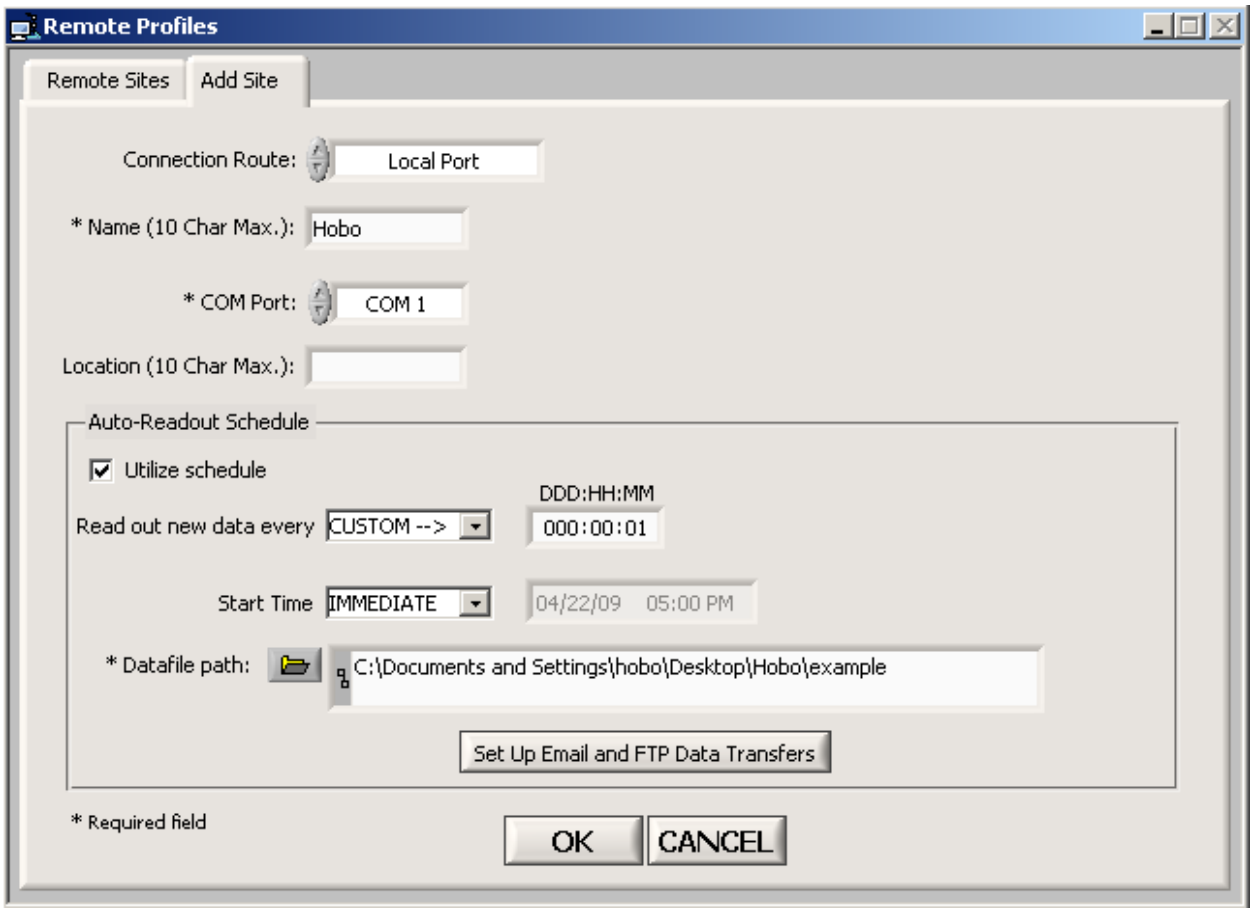


Figure 7 – Remote Site Manager software Add Site page

At this screen, as shown in Figure 7, you will need to give the site a name where it says “* Name”. Then from the choices for “Connection Route” select “Local Port”. Now where it says “* COM Port” select “COM 1” or which serial port that you have connected the radio module to. The Auto-Readout Schedule will need to be setup in order for the program to auto readout. To setup the auto readout select the checkbox next to “Utilize Schedule”. Next change the “Read out new data every” and select “CUSTOM”. This will allow you to select how often you would like it to sample in the column “DDD:HH:MM”. Change this value to 3 minutes. Now where it says “* Datafile path” select where you would like to save the files containing the sensor data that this software is generating. Once you have these parameters filled out click “OK” button. From this screen click “Connect” and a screen like Figure 8 will appear. This page is attempting to connect to the data logger and it should connect if the parameters are correct.

Figure 8 – Remote Connection page from Remote Site Manager

If the parameters are not correct the software will tell you that it is not connected. If this is the case, edit your parameters to the correct ones. Once you have the remote site connected to, close the screen. Now you will be back at a screen shown in Figure 6. Here you should see the site you just created in the "Auto-Readout Schedule". Click on the "activate" box and select the correct remote site to activate to begin the automatic readout process.

5. Java-Based Updater: HOBO Builder

The HOBOware has no means of populating the database used for this project so a custom application was made. Programmed in Java, HOBO Builder will connect to the database and automatically keep it updated. The program flowchart is shown in Appendix A and its code in Appendix B.

It is recommended that the HOBOBuilder.java file is stored in the location you choose the Onset Remote Site Manager to save the sensor readout files. After the program has been compiled it is launched from the command line console. The proper syntax being (an example is shown in Figure 8):

java HOBOBuilder <update interval (seconds)> <sensor readout .txt file's path>

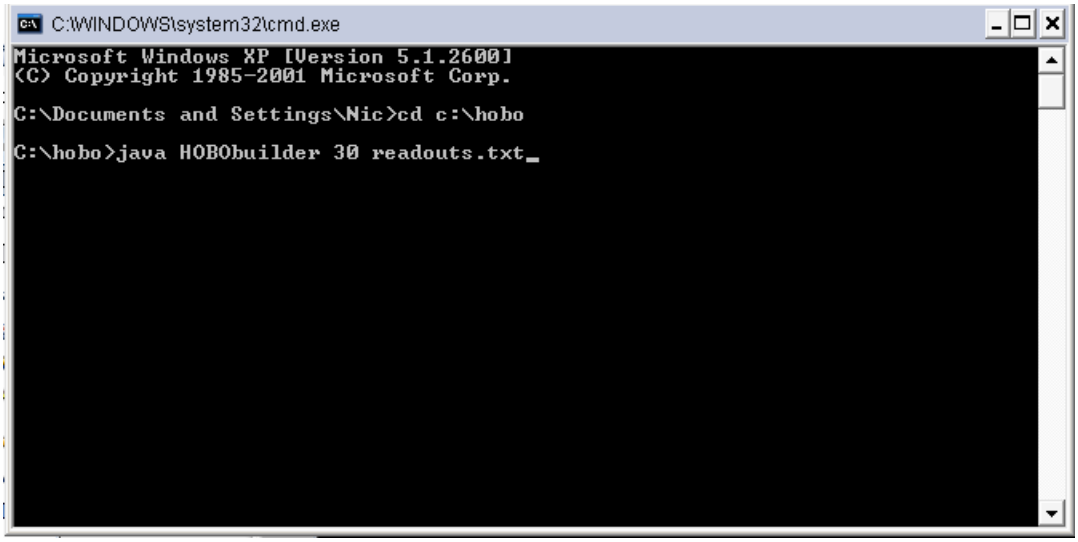


Figure 8 – Launching the HOBO Builder java program.

Once launched the program can run automatically. Upon successful running the screen should look similar to Figure 9.

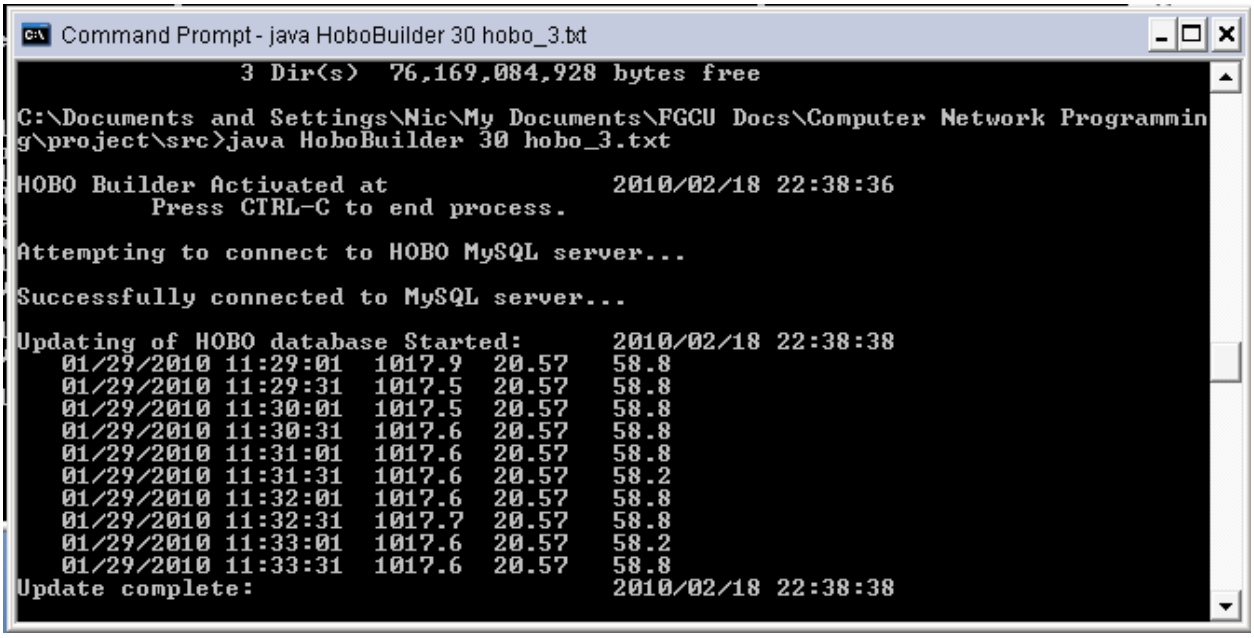


Figure 9 – Successful launch of HOBO Builder.

As one can see the program will connect to the hobo mySQL database on satnet and enter the new readings. These new entries will be displayed in the console window additionally. The

update cycle can be halted by pressing “CTRL-C”. In addition, this action will also quit the program.

6. Database Organization and Connections

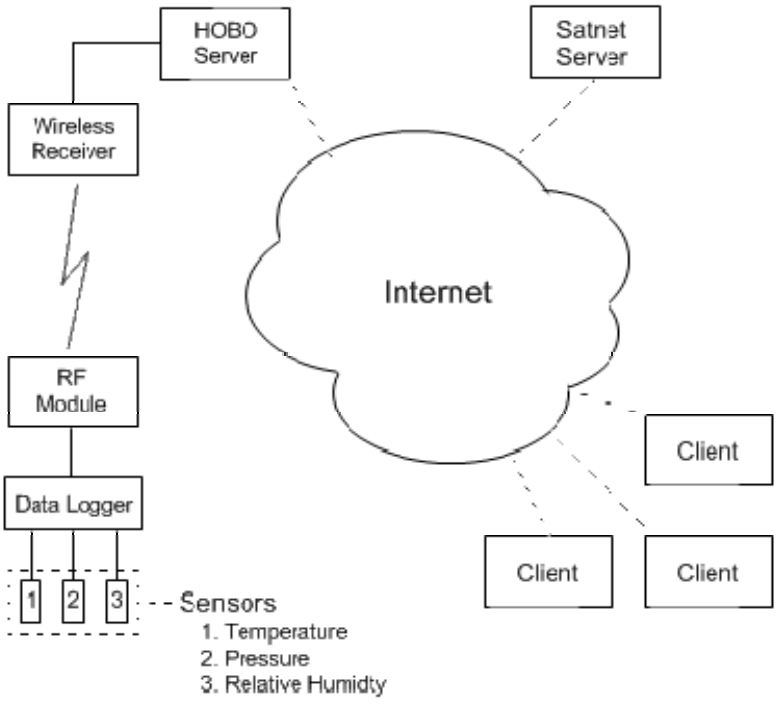


Figure 10- Diagram of the Hardware used.

Figure 10 shows a diagram of how all the hardware involved in the project connects. Data is collected by the sensors and stored on the data logger. The logger will then periodically send the stored readings to the HOBO service via the logger’s wireless system. The data is then stored in a text file on the HOBO Server via the HOBO software, and is then entered into the database on the satnet server using the Java-Based updated design specifically for the project. The data can then be viewed through the internet by clients by accessing the website located on the satnet server.

6.1 Database Structure

The database used is a simple one with a single table, readouts, and is located on the satnet server. This table contains four columns: dateTime, pressure, temperature and rh. Figure 10 shows a description of the readouts table showing the field’s name, type, and other attributes.

Field	Type	Null	Key	Default	Extra
dateTime	timestamp	YES		NULL	
pressure	double	YES		NULL	
temperature	double	YES		NULL	
rh	double	YES		NULL	

Figure 11- Description of the Database

While not shown in Figure 11; dateTime is used a primary key to identify each entry as unique.

In order to manually check the database, the user would have to have an account on the satnet server with the proper privileges granted to use mySQL. After connecting to the server, preferably via SSH for this case, one would use the following simple command to access mySQL:

```
mysql
```

Once connected to mySQL the following command would be used to access the specific database used for this project

```
use hobo;
```

With the database selected the user can now directly interact with it by viewing the makeup of the table…

```
Describe readouts;
```

View entries already in…

```
Select * from readouts;
```

Add entries …

```
Insert into readouts (dateTime,pressure,temperature,rh) values (val1, val2,val3,val4);
```

Remove entries…

```
DELETE FROM readouts WHERE dateTime = dateToBeDeleted;
```

Update entries…

```
Update readouts SET columnName=someValue WHERE someColumn=somevalue;
```

There are more SQL commands to interact with the database but these provide the gist of the desired function.

6.2 Database Connectivity

The database is connected to from two points: HoboBuilder (the Java program that updates the database which is located on the server hosting the HOBO software and hardware) and the website.

In order to connect to a database in Java a driver specific to the implantation of the database must be loaded in the program. In this case the Connector/J MySQL JDBC driver is used. In order to use this driver one must download and install the jar file from www.MySQL.com. Once installed the driver is loaded in the program with the following code (see Appendix C for more detailed code of the Java-based updater):

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

Once loaded a connection can be established by adding the following line of code:

```
Connection con = DriverManager.getConnection("jdbc:mysql://hostname/DB name",
"user", "password");
```

In order for these two lines of code to work the java.sql package must be imported.

To connect to the database via the website functions within PHP are used. The two following lines of PHP code are used to connect to a mySQL database (see appendix E for more detailed code of the website):

```
$conn = mySQL_connect($host, $dbuser, $dbpass) or die ('Error');
MySQL_select_db($dbname);
```

Both the mySQL_connect() and mySQL_select_db() functions are built into PHP provided mySQL is enabled in the php.ini. The values proceeded by a $ are variables in PHP that the coder would have set up.

Once the connections are setup SQL statements are passed to them to query the database. This passing is done with PHP's mysql_query() function, for example:

```
$query = "SELECT * FROM readouts ORDER BY dateTime DESC";
```

```
$result = mysql_query($query);
```

In this case, the SQL statement was saved to a local variable first before passing, due to the author's preference. The function will return a return a ResultSet that contains the rows in the database that satisfied the query; for this example that would be all entries and sorting them by dateTime in descending order. Another example would be:

```
$query = "SELECT * FROM readouts WHERE $filter BETWEEN $low AND $high ORDER BY dateTime DESC";
$result = mysql_query($query);
```

This time the results returned would be more specific. $filter would have retrieved from a list box containing which column the user wants to filter the results by. $low and $high would also be retrieved from a user inputted value and be used to select entries which fall between these two values.

7. Full Feature Website

7.1 Home Page

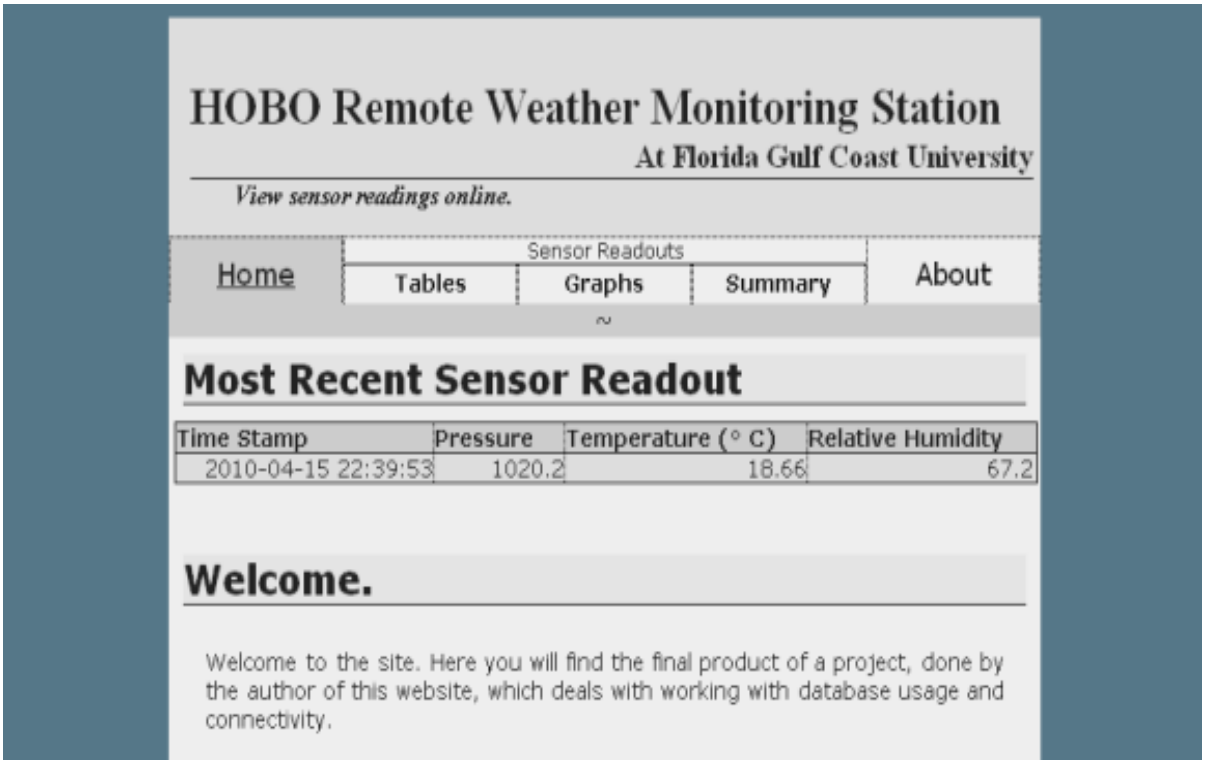


Figure 11 - Index Page of Website

Figure 14 shows a screenshot of the home page, index.php. At the top you can see the header which displays the name of the site. Below it is the menu. Both of these items will on every page of the site.

When the mouse hovers over a button on the menu, that buttons background color will darken. As one can see in Figure 14, the button of the current page is further darkened and disabled from being clicked on. The following code shows how the menu is created:

```
<div id="menu">
      <div id="mDBtn">
       <div id="menuTxt"> <u> Home </u> </div>
      </div>

      <div id="menuSub">
        <div id="menuSens"> Sensor Readouts </div>

        <br />
        <div id="mSBtn" onclick="window.location.href='tables.php?page=1'">
         <div id="menuTxtE"> <a href="tables.php?page=1"> Tables </a> </div>
        </div>

        <div id="mSBtn" onclick="window.location.href='graphs.php'">
         <div id="menuTxtE"> <a href="graphs.php"> Graphs </a> </div>
        </div>

        <div id="mSBtn" onclick="window.location.href='summary.php'">
         <div id="menuTxtE"> <a href="summary.php"> Summary </a> </div>
        </div>
      </div>

      <div id="mSBtn2" onclick="window.location.href='about.html'">
        <div id="menuTxt"> <a href="about.html"> About </a> </div>
      </div>
</div>
```

Below the menu, the most recent entry will be displayed. The following code shows how:

```
<?php
      $dbhost = 'satnet.fgcu.edu';
      $dbuser = 'nmlv2';
      $dbpass = '';

      $conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error
connecting to mysql');

      $dbname = 'hobo';
      mysql_select_db($dbname);

      $query = "SELECT * FROM readouts ORDER BY dateTime DESC LIMIT 1";

            $result = mysql_query($query);
            $num=mysql_numrows($result);

            $dateTime = "";
          if($result) {
            while($row = mysql_fetch_assoc($result)) {
```

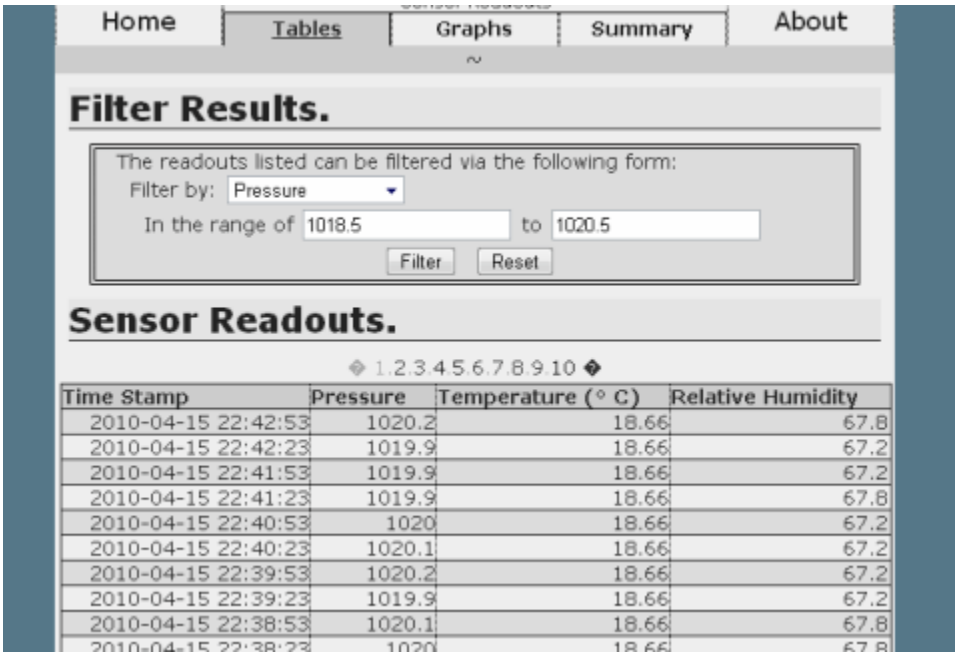
```
      $dateTime=$row["dateTime"];
      $pressure=$row["pressure"];
      $temperature=$row["temperature"];
      $rh=$row["rh"];
    }
  }

    echo "<div id=\"dbHeader\">";
    echo "  <div id=\"dbTimeCol\"> Time Stamp </div>";
    echo "  <div id=\"dbPresCol\"> Pressure </div>";
    echo "  <div id=\"dbTempCol\"> Temperature (&deg; C) </div>";
    echo "  <div id=\"dbRHCol\"> Relative Humidity </div>";
    echo "</div>";

    if($dateTime!="") {
      echo "<div id=\"dbRow0\">";
      echo "  <div id=\"dbTimeCol\"> $dateTime </div>";
      echo "  <div id=\"dbPresCol\"> $pressure </div>";
      echo "  <div id=\"dbTempCol\"> $temperature </div>";
      echo "  <div id=\"dbRHCol\"> $rh </div>";
      echo "</div>";
  } else {
    echo "<div id=\"dbRow0\">";
    echo "Database empty.";
    echo "</div>";
  }
?>
```

Following that will just be headers and paragraphs of general information about the site.

7.2 Tables Page



Time Stamp	Pressure	Temperature (° C)	Relative Humidity
2010-04-15 22:42:53	1020.2	18.66	67.8
2010-04-15 22:42:23	1019.9	18.66	67.2
2010-04-15 22:41:53	1019.9	18.66	67.2
2010-04-15 22:41:23	1019.9	18.66	67.8
2010-04-15 22:40:53	1020	18.66	67.2
2010-04-15 22:40:23	1020.1	18.66	67.2
2010-04-15 22:39:53	1020.2	18.66	67.2
2010-04-15 22:39:23	1019.9	18.66	67.2
2010-04-15 22:38:53	1020.1	18.66	67.8

Figure 12- Screenshot of Tables Page.

Figure 15 is a screenshot of the tables page. Under the menu a form is placed to allow the filtering of the results. The following code shows how:

```
<form id="reg" action="tables.php" method="get">
    <div id="frmText">
          The readouts listed can be filtered via the following form:
        </div>

        <input type="hidden" name="page" value="1" />

        <br />

          <div id="frmFields">
          Filter by:
          <select class="field" name="selType" id="selType"
onchange="Javascript:CheckDisplay()">
            <option value="All">All</option>
            <option value="dateTime">Time Stamp</option>
            <option value="pressure">Pressure</option>
            <option value="temperature">Temperature</option>
            <option value="rh">Relative Humidity</option>
          </select>
        </div>

        <br />

        <div id="dbHidden" style="display:none">
          In the range of
          <input type="text" id="txtLow" name="txtLow" size="19"
maxlength="19" />
          to
          <input type="text" id="txtHigh" name="txtHigh" size="19"
maxlength="19" />
        </div>

        <br />

        <div id="frmBtns">
          <input type="button" id="btnFilter" value="Filter"
onmousedown="Javascript:Filter()" /> 
          <input type="reset" id="btnReset" value="Reset" />
        </div>
      </form>
```

The hidden type of name page is there to allow the form to work with the pagination code as the code requires that a page value be passed with the form. The div, dbHidden, is only shown if the filter type is not set to all.

Below the form the actual data collected from the HOBO unit is displayed. Due to the amount of possible entries the results are broken up by pagination. This makes it so that no more than 300 entries are shown at once. The user can simply click the page numbers above the table to see the

rest of the entries. The code for this is too long for a meaningful snippet, so refer to Appendix E for details.

7.3 Graph Page

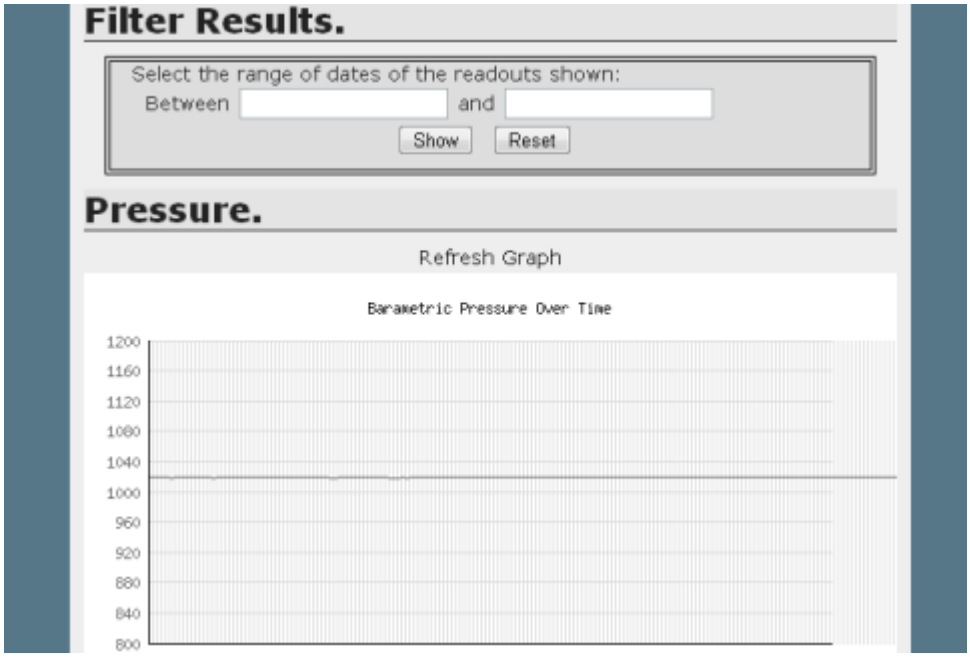


Figure 13- Screenshot of Graph Page.

Figure 16 shows a screenshot of the graphing page. This page also contains a form to filter the results shown but this one only accepts dates and times to limit. Under the form is a button to refresh the graph below. The button will only refresh the one graph and not affect the others or reload the page. The code to handle this is too long for a meaningful snippet, so refer to Appendix D for more details.

Under the refresh button is the graph itself. There are three graphs (only one is shown in the screenshot), one for each of the sensor types currently attached to the HOBO, and each has its own refresh button. The graph is created by the PHPGraphLib package. This is done by passing a array of data to the library which will return a PNG image file. The following code snippet shows an example;

```
   $sql="SELECT dateTime, pressure FROM readouts WHERE dateTime BETWEEN '$low'
AND '$high'";
   $result = mysql_query($sql);

     if($result) {
        while($row = mysql_fetch_assoc($result)) {
         $dateTime=$row["dateTime"];
         $pressure=$row["pressure"];
         //ADD TO ARRAY
         $dataArray[$dateTime]=$pressure;
        }
     }
     $graph->addData($dataArray);
     $graph->setTitle("Barametric Pressure Over Time");
     $graph->setRange(1200,800);
     $graph->setBars(false);
     $graph->setLine(true);
     $graph->createGraph();
```

7.4 Summary Page

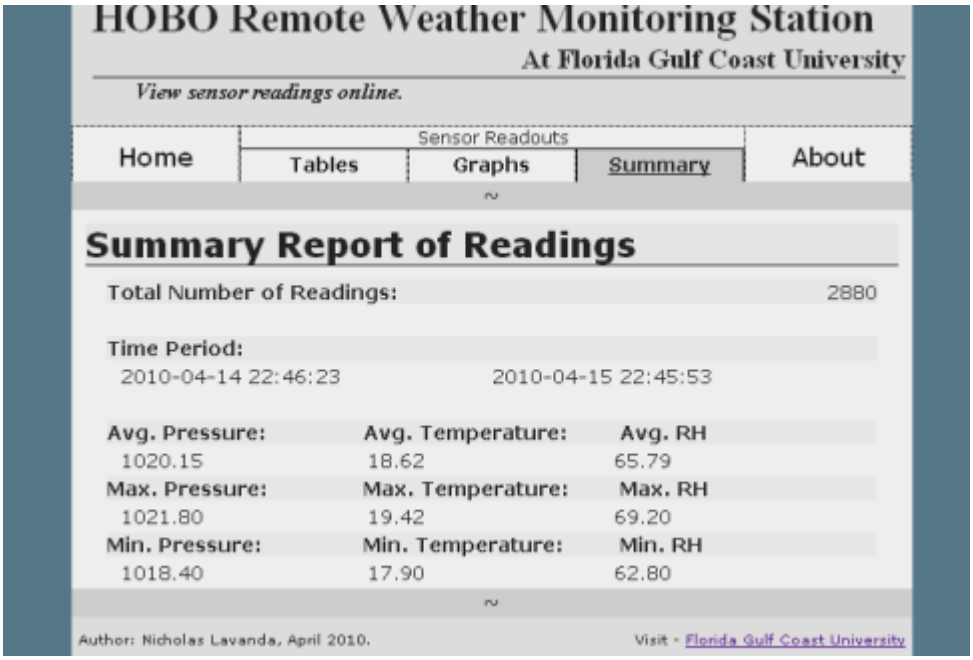


Figure 14- Screenshot of the Summary page.

Figure 17 shows a screenshot of the summary report page. Below the menu the report is made. The data is collected by one large SQL statement.

```
   $query = "SELECT COUNT(*) AS numRead, AVG(pressure) AS avgPres,
AVG(temperature) AS avgTemp, AVG(rh) AS avgRH, MAX(pressure) AS maxPres,
MAX(temperature) AS maxTemp, MAX(rh) AS maxRH, MIN(pressure) AS minPres,
MIN(temperature) AS minTemp, MIN(rh) AS minRH, MIN(dateTime) AS minTS,
MAX(dateTime) AS maxTS FROM readouts";

           $result = mysql_query($query);

           if($result) {
             while($row = mysql_fetch_assoc($result)) {
               $numRe=$row["numRead"];
             $avgPr=$row["avgPres"];
                   $avgTe=$row["avgTemp"];
               $avgRh=$row["avgRH"];
               $maxPr=$row["maxPres"];
               $maxTe=$row["maxTemp"];
               $maxRh=$row["maxRH"];
               $minPr=$row["minPres"];
               $minTe=$row["minTemp"];
               $minRh=$row["minRH"];
               $minTS=$row["minTS"];
               $maxTS=$row["maxTS"];
             }
           }
```

Then the results are simply displayed using headers and paragraphs. Also at the bottom the page on can see a footer. This footer is also on every page.

7.5 About Page

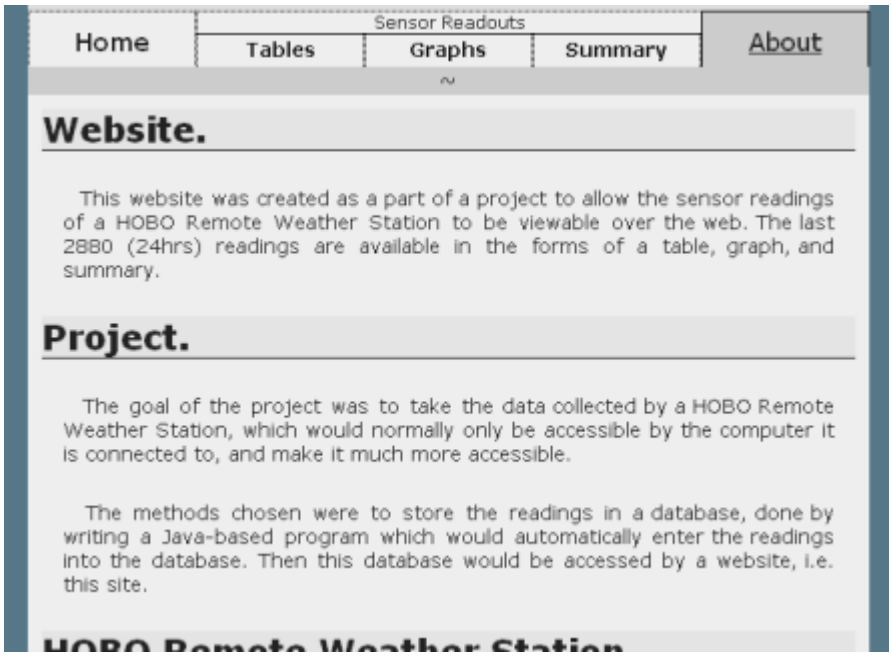

Home
Sensor Readouts
Tables
Graphs
Summary
About
Website.
This website was created as a part of a project to allow the sensor readings of a HOBO Remote Weather Station to be viewable over the web. The last 2880 (24hrs) readings are available in the forms of a table, graph, and summary.
Project.
The goal of the project was to take the data collected by a HOBO Remote Weather Station, which would normally only be accessible by the computer it is connected to, and make it much more accessible.
The methods chosen were to store the readings in a database, done by writing a Java-based program which would automatically enter the readings into the database. Then this database would be accessed by a website, i.e. this site.

Figure 15- Screenshot of the about page.

Figure 18 shows a screenshot of the About page. This page just displays some information about the site, project, equipment used, and author. This is done just using simple headers and paragraphs.

8. Mobile Website

8.1 Index

The mobile website is relatively easy to comprehend and navigate. The site is divided up into five pages with the first being the index. The index simply lists the sensors the user can access as well as an about page. These actions are diagrammed below in Figure 16.

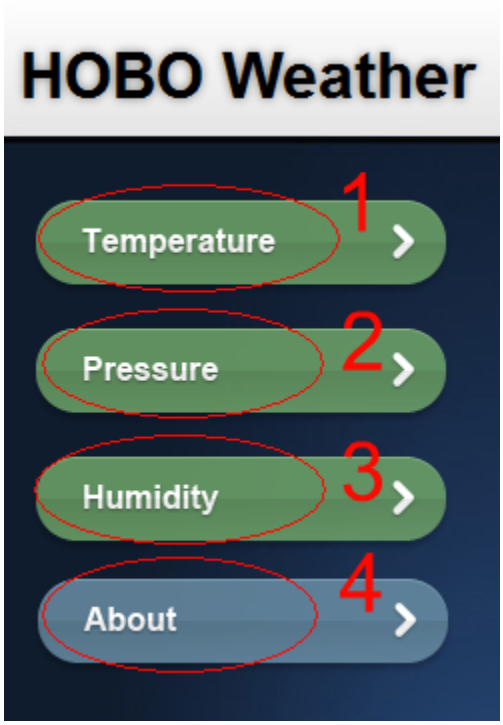


Figure 16 - Mobile Index

1 - Selecting the temperature button will take the user to the temperature mobile web page

2 - Selecting the pressure button will take the user to the pressure mobile web page

3 - Selecting the humidity button will take the user to the humidity mobile web page

4 - Selecting the about button will take the user to the about mobile web page

8.2 Sensor Pages

All the sensor pages are programmed very similarly and will be described together. Each page contains a graphical display for the data requested. The user can only perform one action on these pages, which is clicking the back button (symbolized by a green arrow - shown in Figure 17). This keeps the site simple and intuitive.

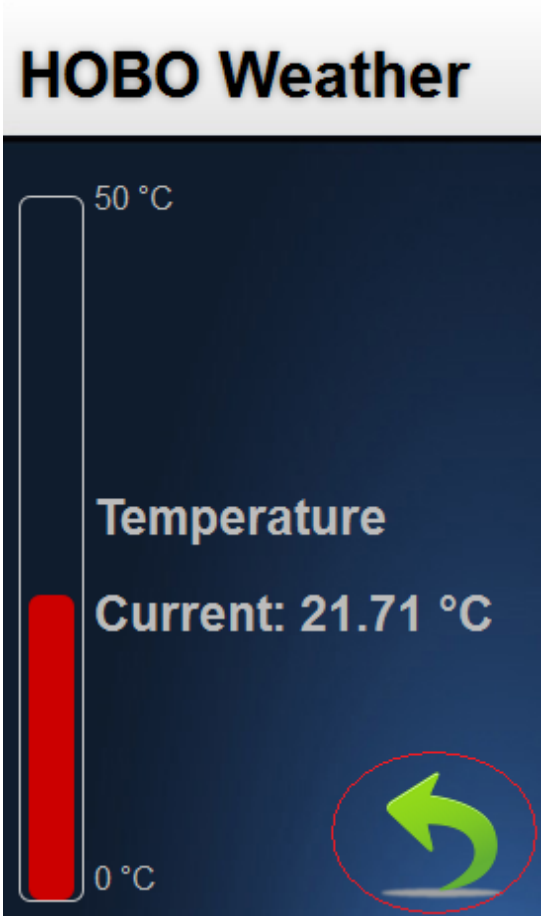


Figure 17 - Mobile Sensor

8.3 About

The about page simply states the author of the site and provides a link to the full feature site developed in the previous iteration of the project. Here there are two user interactions possible, selecting the aforementioned link the the full feature site or selecting the back button as on the sensor pages.

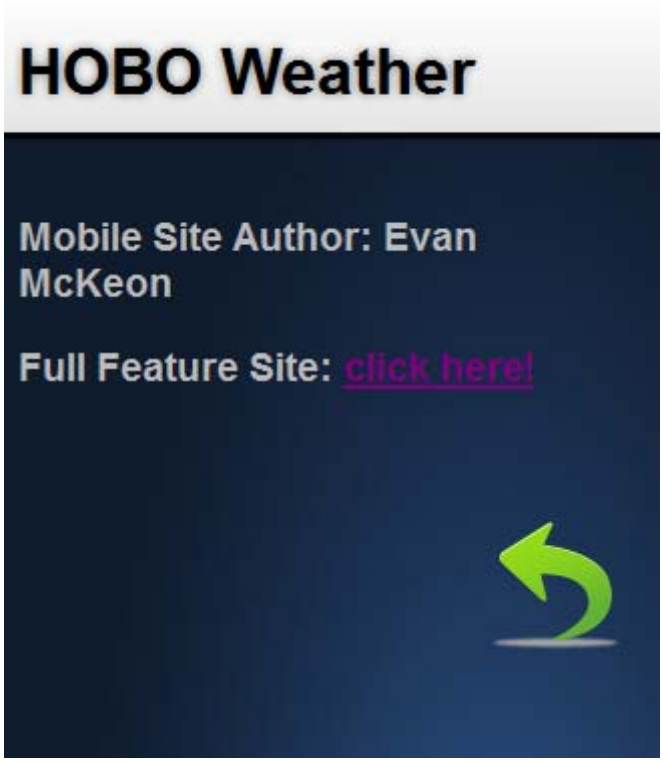


Figure 18 - Mobile About

Appendix A: Troubleshooting

A1. HOBO Configuration

Problem: Hobo Hardware is not being recognized by HOBOware.

Solution: Sometimes, usually after extended non-use, the software may have difficulty detecting the hardware. This can generally be fixed with a combination of unplugging/plugging connections and removing and reinserting batteries or other power sources, much like a resetting a home router. If the hardware still fails to be recognized then the problem likely lies with a dead power source or faulty hardware.

Problem: Hobo Hardware is not being recognized by Onset Remote Manager.

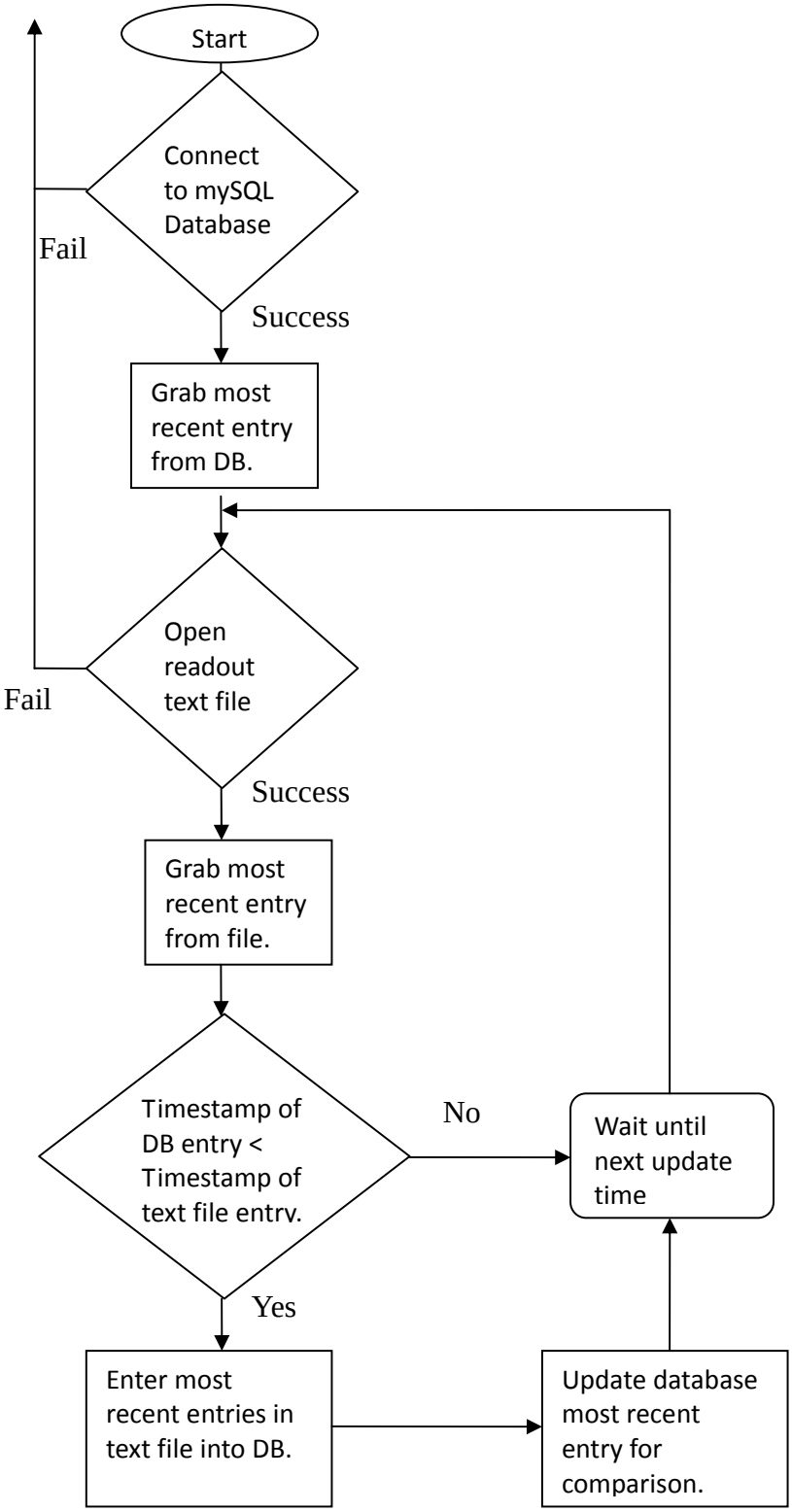
Solution: If the "OK" light on the data logger is flashing and everything is connected properly and the hardware is still not being detected by Onset Remote Manager then the problem is likely with the settings placed on the screen shown back in Figure 7. The "Connection Route" setting can have problems sometimes. If "Local Port" does work then select the "Remote Modem" option. Simply put, usually only one of this will work, so it may be necessary to try the other if one does not work.

A2. HOBO Builder – Java based Updater

Problem: Console displays: "NULL POINTER: Could not find a entry in the database that matched an entry in the <filename>. This likely means the file passed does not match the original file used to create the database. This program is not designed to handle this case"

Solution: The HOBO Builder software assumes that the readouts will always be saved to the same file. If a file other than the one originally used to create the database is used then the software will fail to update the database as it will find no entries in the new readouts file which already exists in the database. The prevents the program from finding a reference point on where to start the update.

Appendix B: HOBO Builder Program Flowchart



Appendix C: HoboBuilder.java

```
/**
 * @(#)HoboBuilder.java
 *
 *
 * @author Nicholas Lavanda
 * @version 1.00 2009/10/7
 *
 *     Reads the text file generated by the HOBOware software. The text file contains
 *     a log of the various sensor readings. This version can handle a HOBO unit that
 *     has Barometic Pressure, Temperature, and Relative humidity sensors and the unit
 *     must have all three, no less, no more. This data is then entered into a
 *     database, no duplicate entries.
 *
 *     The user must determine the time inbetween the reads and enters via a command
 *     line parameter.
 *
 *  NOTE: This program does handle the cases where the database is modifed by
 *  another source while running, specifically deletion of records in the database.
 *  If records are deleted or the entire database is empty during runtime the
 *  program will not catch and continue as if the database was in the same state
 *  as the last update.
 */

import java.sql.*;
import java.io.*;
import java.util.*;
import java.text.SimpleDateFormat;
import java.text.DateFormat;

public class HoboBuilder {
```

```
    private static boolean active;


public HoboBuilder() {}


public static void main(String [] args) throws SQLException {


    String input = null;
    String fileName = "";
    SimpleDateFormat gFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
    java.sql.Timestamp recentEntry = null;
    java.sql.Timestamp dateTime = null;
    Double pressure;
            Double temperature;
            Double rh;
            long sleepTime = 0;
            int entryCount = 0;
            Connection con = null;
            Statement stmt = null;
            boolean emptyDB = false;




            // Check to see if the sleep time was provided. Exit if not.
            if (args.length!=2) {
                    System.out.println("Usage: HoboBuilder <sleep time between updates in
seconds> <hobo file>");
                    System.exit(1);
            } else {
                    fileName = args[1];
                    sleepTime = Long.parseLong(args[0]);


                    // Check to see if a long enough time has been given.
                    // While the HOBO software can be set to readout every second the amount
```

```
of time
                        // it actually takes to complete the readout will around 30 secs or more.
                        if (sleepTime<30) {
                                System.out.println("\nUpdate Cycle must be greater than 30s as
that is the shortest time between readings on HOBO unit itself.");
                                System.exit(1);
                        }
                        // User input in seconds, convert to milliseconds.
                        sleepTime *= 1000;

                        // Notify that the HOBO Builder process has begun by outputting to
console.
                        System.out.println("\nHOBO Builder Activated at \t\t" + timeStamp() +
"\n\t Press CTRL-C to end process.\n");
                        active=true;
                }

                /**********************************************************************/
                //connect to database
                try {
                        System.out.println("Attempting to connect to HOBO MySQL server...\n");

                Class.forName("com.mysql.jdbc.Driver").newInstance();
                con = DriverManager.getConnection("jdbc:mysql://satnet.fgcu.edu/hobo", "nmlv2",
"");

                if(!con.isClosed())
                        System.out.println("Successfully connected to MySQL server...\n");

                stmt = con.createStatement();

        } catch(Exception e) {
                System.err.println("Exception: " + e.getMessage() + "\n");
                System.out.println("Connection to MySQL server failed. Exiting program.");
```

```
                System.exit(1);
        }

                /*************************************************************************/
                // get the total amount of entries currently in the db
                ResultSet rsCount = stmt.executeQuery("SELECT COUNT(*) as count FROM readouts");
                rsCount.next();
                entryCount = rsCount.getInt("count");

                //grab most recent entry
                ResultSet rs = stmt.executeQuery("SELECT dateTime FROM readouts ORDER BY
dateTime");
                rs.next();

                /*to prevent a possible overload on the satnet servers the ammount of entries in
the
                * database is limited to 2880 entries or 24 hours worth of data at a 30 sec
sensor
                * reading interval
                */
                if (entryCount>2880) {
                        // grab the first entry
                        java.sql.Timestamp initial = rs.getTimestamp("dateTime");

                        // skip ahead the (total numberof entry - 2880)th entry in the db
                        int toDelete = entryCount - 2880;
                        for (int i = toDelete; i>0; i--)
                                rs.next();

                        // grab the (total numberof entry - 2880)th entry
                        recentEntry = rs.getTimestamp("dateTime");

                        //delete the entries from the inital date to the (total numberof entry -
2880)th date
                        // to reduce the number of entries to 2880
```

```
                        stmt.executeUpdate("DELETE FROM readouts WHERE dateTime BETWEEN \"" +
initial + "\" AND \"" + recentEntry + "\"");

                        entryCount = 2880;

                }


                rs = stmt.executeQuery("SELECT dateTime FROM readouts ORDER BY dateTime DESC
LIMIT 1");

                if (rs.next())

                        recentEntry = rs.getTimestamp("dateTime"); // the most recent entry




                do {

                        //Notify that reading has begun by outputting to console.

                        System.out.println("Updating of HOBO database Started: \t" + timeStamp());


                        try {

                                // Open the text file containing the HOBO read out (hobo.txt).

                                BufferedReader in = new BufferedReader (new FileReader(fileName));


                                // Scroll past the information not needed in hobo.txt

                                for (int i=0;i<5;i++)

                                        input = in.readLine();


                                // Scroll past the the entries already in database

                                try {

                                        do {

                                                input = in.readLine();


                                                //Parse the date and time from the string

                                                StringTokenizer st = new StringTokenizer(input,
"\t");

                                                String dateStr = st.nextToken();

                                                String dateStrF = gFormat.format(new
java.util.Date(dateStr));
```

```
                                        dateTime = new
java.sql.Timestamp((gFormat.parse(dateStrF)).getTime());


                                        // If no entries in database then use first in
hobo.txt.
                                        if (recentEntry==null) {
                                                recentEntry = dateTime;
                                                emptyDB = true;
                                                break;
                                        }


                                } while(recentEntry.after(dateTime));
                        } catch (NullPointerException e) {
                                System.out.println("\nNULL POINTER: Could not find a entry
in the database that " +
                                        "matched an entry in the \"" + fileName + "\". This
likely means the file passed " +
                                        "does not match the orginal file used to
create the database. This program " +
                                                "is not designed to handle this
case");
                                System.exit(1);
                        }


                        // if at end of file, data base is up to date
                        //              close
                        // else, new entries to be input
                        //              parse and place into databse
                        if (!in.ready()) {
                                System.out.println("Database is already up to date.");
                                in.close();

        /*************************************************************/
                                // Close Connection to database
                                try {
                                if(con != null)
```

```
                                        con.close();
                                } catch(SQLException e) {}
                                } else {

        /************************************************************/
                                        // Check to see if database is open
                                        if(con.isClosed())
                                                try {
                                                        System.out.println("Attempting to connect to
HOBO MySQL server...\n");

        Class.forName("com.mysql.jdbc.Driver").newInstance();
                                                con =
DriverManager.getConnection("jdbc:mysql://satnet.fgcu.edu/hobo", "nmlv2", "");

                                                if(!con.isClosed())
                                                        System.out.println("Successfully connected
to MySQL server...\n");

                                                stmt = con.createStatement();

                                        } catch(Exception e) {
                                                System.err.println("Exception: " + e.getMessage() +
"\n");
                                                System.out.println("Connection to MySQL server
failed. Exiting program.");

                                                        System.exit(1);
                                        }

                                        while(in.ready()) {

                                                // If the database is empty dont refresh input, use
current.
                                                if (!emptyDB) {
                                                        input = in.readLine();
```

```
                                        }
                                        System.out.println("   " + input);

                                        //Parse the date and time from the string
                                        StringTokenizer st = new StringTokenizer(input,
"\t");
                                        String dateStr = st.nextToken();
                                        String dateStrF = gFormat.format(new
java.util.Date(dateStr));

                                        dateTime = new
java.sql.Timestamp((gFormat.parse(dateStrF)).getTime());

                                        //Parse the pressure from the string
                                        pressure=Double.parseDouble(st.nextToken());

                                        //Parse the temperature from the string
                                        temperature=Double.parseDouble(st.nextToken());

                                        //Parse the relative humidity from the string
                                        rh=Double.parseDouble(st.nextToken());



      /*************************************************************
                                        * Delete old entries to make room for new ones
                                        */
                                        if (entryCount>2879) {
                                                //get the oldest entry
                                                ResultSet rs2 = stmt.executeQuery("SELECT
dateTime FROM readouts ORDER BY dateTime LIMIT 1");

                                                // date of the oldest entry
                                                rs2.next();
                                                java.sql.Timestamp victim =
rs2.getTimestamp("dateTime");
```

```
                                                //remove the oldest entry
                                                stmt.executeUpdate("DELETE FROM readouts
WHERE dateTime = \"" + victim + "\"");
                                                entryCount--;
                                        }


        /*************************************************************/
                                        // Enter Parses into database
                                        stmt.executeUpdate("INSERT INTO readouts ( dateTime,
pressure, temperature, rh ) VALUES ( \""+ dateTime + "\", "+ pressure+ ", "+temperature+",
"+rh+" )");
                                        entryCount++;
                                        emptyDB=false;
                                }


                                //Update the time of the most recent entry.
                                recentEntry = dateTime;


                                // Close hobo.txt
                                in.close();


                                //Notify that reading has ended by outputting to console.
                                System.out.println("Update complete: \t\t\t" +
timeStamp());


        /*************************************************************/
                                // Close Connection to database
                                try {
                                if(con != null)
                                        con.close();
                        } catch(SQLException e) {}
                        }
```

```
                } catch (Exception e) {
                        System.out.println(e);
                }

                // Wait until next check.
                try {
                        Thread.sleep(sleepTime);
                } catch(Exception e) {
                        System.out.println(e);
                }
        }while(active);
    }

    //Returns the current time and date
    //
    public static String timeStamp() {
        java.util.Date readStamp = new java.util.Date();
        SimpleDateFormat formatter = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
        return formatter.format(readStamp);
    }
}
```

Appendix D: hobo.js – Form parameter editor

```
function Filter() {
	if (trim(document.getElementById('selType').value) == "All") {
	  SendPost();
	} else if (trim(document.getElementById('selType').value) == "dateTime") {
	  if (trim(document.getElementById('txtLow').value) == "" &&
trim(document.getElementById('txtHigh').value) == "") {
		alert("Please enter a range of values to filter.");
		document.getElementById('txtLow').focus();
	  } else {
		CheckTSLow();
	  }
	} else {
	  if (trim(document.getElementById('txtLow').value) == "" &&
trim(document.getElementById('txtHigh').value) == "") {
		alert("Please enter a range of values to filter.");
		document.getElementById('txtLow').focus();
	  } else {
		CheckLow();
	  }
	}
}

function CheckLow() {
	if   (isNaN(document.getElementById('txtLow').value)) {
		alert("Please insert a numerical value for the low range.");
		document.getElementById('txtLow').focus();
	} else {
		CheckHigh();
	}
}

function CheckHigh() {
	if   (isNaN(document.getElementById('txtHigh').value)) {
		alert("Please insert a numerical value for the high range.");
		document.getElementById('txtHigh').focus();
	} else {
		SendPost();
	}
}


function CheckTSLow() {
	var reTS = /(\d{2})\/(\d{2})\/(\d{4}) (\d{2}):(\d{2}):(\d{2})/;

	if   (!reTS.test((document.getElementById('txtLow').value)) ||
trim(document.getElementById('txtLow').value) == "") {
		alert("Time stamp low range must be in the format of DD/MM/YYYY HH:MM:SS");
		document.getElementById('txtLow').focus();
	} else {
		CheckTSHigh();
	}
}

function CheckTSHigh() {
	var reTS = /(\d{2})\/(\d{2})\/(\d{4}) (\d{2}):(\d{2}):(\d{2})/;

	if   (!reTS.test((document.getElementById('txtHigh').value)) ||
trim(document.getElementById('txtHigh').value) == "") {
		alert("Time stamp high range must be in the format of DD/MM/YYYY HH:MM:SS");
		document.getElementById('txtHigh').focus();
	} else {
		SendPost();
	}
}


function SendPost() {
	document.getElementById('reg').submit();
}
//-----------------------------------------------------------------------------------------------
----------------------------
function FilterGraph() {
	var count = 0;
```

```
	if (trim(document.getElementById('txtLow').value) != "") {
		count += CheckTSLowG();
	}
	if (trim(document.getElementById('txtHigh').value) != "") {
		count += CheckTSHighG();
	}

	if (count==0) {
		SendPost();
	}
}

function CheckTSLowG() {
	var reTS = /(\d{2})\/(\d{2})\/(\d{4}) (\d{2}):(\d{2}):(\d{2})/;

	if   (!reTS.test((document.getElementById('txtLow').value))) {
		alert("Time stamp low range must be in the format of DD/MM/YYYY HH:MM:SS");
		document.getElementById('txtLow').focus();
		return 1;
	} else {
		return 0;
	}
}

function CheckTSHighG() {
	var reTS = /(\d{2})\/(\d{2})\/(\d{4}) (\d{2}):(\d{2}):(\d{2})/;

	if   (!reTS.test((document.getElementById('txtHigh').value))) {
		alert("Time stamp high range must be in the format of DD/MM/YYYY HH:MM:SS");
		document.getElementById('txtHigh').focus();
		return 1;
	} else {
		return 0;
	}
}

//-------------------------------------------------------------------------------------------
----------------------------
function CheckDisplay() {

	if (document.getElementById('selType').value == "All") {
		document.getElementById('divID').style.display="none";
	} else {
		document.getElementById('divID').style.display="";
	}

}

//-------------------------------------------------------------------------------------------
----------------------------
function trim(str) {
	return str.replace(/^\s\s*/, '').replace(/\s\s*$/, '');
}
```

Appendix E – view.php – Displays results in text form.

```
<html>
<head>
       <title> HOBO Weather Montitoring Station - Florida Gulf Coast Univerity </title>

       <script type="text/javascript" src="hobo.js"></script>
</head>

<body>

<form id="reg" action="view.php" method="get">
  <div class="left">
    The readouts listed can be filtered via the following form:  <br />
    Filter by:
  <select class="field" name="selType" id="selType" onchange="Javascript:CheckDisplay()">
    <option value="All">All</option>
    <option value="dateTime">Time Stamp</option>
    <option value="pressure">Pressure</option>
    <option value="temperature">Temperature</option>
    <option value="rh">Relative Humidity</option>
  </select>    <br />

  <div id="divID" style="display:none">
    In the range of
  <input type="text" id="txtLow" name="txtLow" size="19" maxlength="19" />
    to
  <input type="text" id="txtHigh" name="txtHigh" size="19" maxlength="19" />
    <br />
  </div>

  <input type="button" id="btnFilter" value="Filter" onmousedown="Javascript:Filter()" /> 
  <input type="reset" id="btnReset" value="Reset" />
</form>

  <?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = '';

    $conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error connecting to mysql');

    $dbname = 'hobo';
    mysql_select_db($dbname);
  ?>

  <?php
    $filter="";
    if (isset($_GET['selType']))
      $filter = $_GET["selType"];
    if (isset($_GET['txtLow']))
      $low = $_GET["txtLow"];
    if (isset($_GET['txtHigh']))
      $high = $_GET["txtHigh"];

    if ($filter=="All" || $filter=="") {
      $query = "SELECT * FROM readouts ORDER BY dateTime DESC";
    } else if ($filter=="dateTime") {
       $query = "SELECT * FROM readouts WHERE $filter BETWEEN '$low' AND '$high' ORDER BY
dateTime DESC";
    } else {
      $query = "SELECT * FROM readouts WHERE $filter BETWEEN $low AND $high ORDER BY dateTime
DESC";
    }
    $result = mysql_query($query);
    $num=mysql_numrows($result);

    $i=0;
    echo "<h3> Current Weather Station Sensor Readings</h3>";
    echo "<table>";
    echo "<tr><td width=20%><b> Time Stamp </b></td><td width=20%><b> Pressure </b></td><td
width=20%><b> Temperature (&deg; C)</b></td><td width=20%><b> Relative Humidity </b></td>
</tr>";
    while ($i < $num) {
       $dateTime=mysql_result($result,$i,"dateTime");
```

```
        $pressure=mysql_result($result,$i,"pressure");
        $temperature=mysql_result($result,$i,"temperature");
        $rh=mysql_result($result,$i,"rh");

        echo "<tr align=\"right\"> <td> $dateTime </td><td> $pressure </td><td> $temperature
</td><td> $rh </td> </tr>";;
        $i++;
    }
    echo "</table>"
  ?>

</body>
</html>
```

Appendix F – graph.php – Displays results in graph form.

```
<html>
<head>
	<title> HOBO Weather Montitoring Station - Florida Gulf Coast Univerity </title>

	<script type="text/javascript" src="hobo.js"></script>
</head>

<body>
<form id="reg" action="graph.php" method="get">
  <div class="left">
    Select date range of readouts shown:  <br />
      Between
  <input type="text" id="txtLow" name="txtLow" size="19" maxlength="19" />
    and
  <input type="text" id="txtHigh" name="txtHigh" size="19" maxlength="19" />
    <br />

  <input type="button" id="btnShow" value="Show" onmousedown="Javascript:FilterGraph()" /> 
  <input type="reset" id="btnClear" value="Clear" />
</form>

<?php
  $low = "";
  $high = "";

  if (isset($_GET['txtLow']))
    $low = $_GET["txtLow"];
  if (isset($_GET['txtHigh']))
    $high = $_GET["txtHigh"];

  echo "<div><img src=\"pgraph.php?txtLow=$low&txtHigh=$high\" /></div>";
  echo "<div><img src=\"tgraph.php?txtLow=$low&txtHigh=$high\" /></div>";
  echo "<div><img src=\"hgraph.php?txtLow=$low&txtHigh=$high\" /></div>";
  echo "<div align=\"right\"><a href=\"http://www.ebrueggeman.com/phpgraphlib\"> Graphs
generated by PHPGraphLib. </a></div>";
?>
```

Appendix G – summary.php – Displays summary report.

```
<html>
<head>
       <title> HOBO Weather Montitoring Station - Florida Gulf Coast Univerity </title>

</head>

<body>


  <?php
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = '';

    $conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error connecting to mysql');

    $dbname = 'hobo';
    mysql_select_db($dbname);
  ?>

  <?php
    $query = "SELECT COUNT(*) AS numRead, AVG(pressure) AS avgPres, AVG(temperature) AS avgTemp,
AVG(rh) AS avgRH, MAX(pressure) AS maxPres, MAX(temperature) AS maxTemp, MAX(rh) AS maxRH,
MIN(pressure) AS minPres, MIN(temperature) AS minTemp, MIN(rh) AS minRH, MIN(dateTime) AS minTS,
MAX(dateTime) AS maxTS FROM readouts";

    $result = mysql_query($query);

    if($result) {
      while($row = mysql_fetch_assoc($result)) {
        $numRe=$row["numRead"];
       $avgPr=$row["avgPres"];
       $avgTe=$row["avgTemp"];
        $avgRh=$row["avgRH"];
        $maxPr=$row["maxPres"];
        $maxTe=$row["maxTemp"];
        $maxRh=$row["maxRH"];
        $minPr=$row["minPres"];
        $minTe=$row["minTemp"];
        $minRh=$row["minRH"];
        $minTS=$row["minTS"];
        $maxTS=$row["maxTS"];
      }
    }

    echo "<h3> Summary Report of Readings</h3>";
    echo "<table>";
    echo "<tr><td width=20%><b> Total Number of Readings: </b></td><td> $numRe </td></tr>";
    echo  "<tr><td  width=20%><b>  Time  Period:  </b></td><td  colspan=2>  $minTS  --  $maxTS
</td></tr>";
    echo "<tr><td> </td></tr>";
    echo  "<tr><td  width=20%><b>  Avg.  Pressure  </b></td><td  width=20%><b>  Avg.  Temperature
</b></td><td width=20%><b> Avg. RH </b></td></tr>";
    echo "<tr><td width=20%> " . sprintf ("%5.2f", $avgPr) . "</td><td width=20%> " . sprintf
("%5.2f", $avgTe) . " </td><td width=20%>" . sprintf ("%5.2f", $avgRh) . " </td></tr>";
    echo  "<tr><td  width=20%><b>  Max.  Pressure  </b></td><td  width=20%><b>  Max.  Temperature
</b></td><td width=20%><b> Max. RH </b></td></tr>";
    echo "<tr><td width=20%> " . sprintf ("%5.2f", $maxPr) . "</td><td width=20%> " . sprintf
("%5.2f", $maxTe) . " </td><td width=20%>" . sprintf ("%5.2f", $maxRh) . " </td></tr>";
    echo  "<tr><td  width=20%><b>  Min.  Pressure  </b></td><td  width=20%><b>  Min.  Temperature
</b></td><td width=20%><b> Min. RH </b></td></tr>";
    echo "<tr><td width=20%> " . sprintf ("%5.2f", $minPr) . "</td><td width=20%> " . sprintf
("%5.2f", $minTe) . " </td><td width=20%>" . sprintf ("%5.2f", $minRh) . " </td></tr>";
    echo "</table>";
  ?>
</body></html>
```

Appendix H - index.php - Home page of mobile site

```
<html>
<head>
<title>HOBO Mobile View</title>
<link href="mobile.css" rel="stylesheet" type="text/css"/>
<meta name="viewport" content="width=320" />

<body>

<div id="header">
     <h1>HOBO Weather</h1>
</div>

<div>
     <a id="Tbutton" href="temperature.php"><img src="Tbutton.png"
/></a><br />
     <a id="Pbutton" href="pressure.php"><img src="Pbuttons.png"
/></a><br />
     <a id="Hbutton" href="humidity.php"><img src="Hbuttons.png"
/></a><br />
     <a id="Abutton" href="about.php"><img src="Abuttons.png" /></a>
</div>

</body>
</html>
```

Appendix I - temperature.php - First sensor page on mobile site

```
<html>
<head>
<link href="mobile.css" rel="stylesheet" type="text/css"/>
<meta name="viewport" content="width=320" />
<body>
<div id="header">
<h1>HOBO Weather</h1>
</div>
<div>

<?php
$dbhost = 'localhost';
$dbuser = 'nmlv2';
$dbpass = '';

$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error connecting to
mysql');

$dbname = 'hobo';
mysql_select_db($dbname);

$query = "SELECT * FROM readouts ORDER BY dateTime DESC LIMIT 1";

$result = mysql_query($query);
$num=mysql_numrows($result);

$dateTime = "";
if($result) {
 while($row = mysql_fetch_assoc($result)) {
       $dateTime=$row["dateTime"];
       $pressure=$row["pressure"];
       $temperature=$row["temperature"];
       $rh=$row["rh"];
       }
}
$current = $temperature;
$goal = 50;
$mercury = ($current/$goal)*100;
?>
<div id="thermometer">
      <div id="temp" style="height:<?php echo $mercury; ?>%;"></div>
</div>
<div id="text">
      <div id="maxTemp">50 &deg;C</div>
      <h1>Temperature</h1>
      <h1>Current: <?php echo $temperature; ?> &deg;C</h1>
      <div id="minTemp">0 &deg;C</div>
      <div id="backArrow"><a href="index.php"><img src="back.png" /></a></div>
</div>


</div>
</body>
</html>
```

Appendix J - pressure.php - Second sensor page on mobile site.

```
<html>
<head>
<link href="mobile.css" rel="stylesheet" type="text/css"/>
<meta name="viewport" content="width=320" />
<body>
<div id="header">
<h1>HOBO Weather</h1>
</div>
<div>

<?php
$dbhost = 'localhost';
$dbuser = 'nmlv2';
$dbpass = '';

$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error connecting to
mysql');

$dbname = 'hobo';
mysql_select_db($dbname);

$query = "SELECT * FROM readouts ORDER BY dateTime DESC LIMIT 1";

$result = mysql_query($query);
$num=mysql_numrows($result);

$dateTime = "";
if($result) {
 while($row = mysql_fetch_assoc($result)) {
       $dateTime=$row["dateTime"];
       $pressure=$row["pressure"];
       $temperature=$row["temperature"];
       $rh=$row["rh"];
       }
}
$current = $pressure;
$goal = 1200;
$mercury = ($current/$goal)*100;
?>
<div id="thermometer">
      <div id="pressure" style="height:<?php echo $mercury; ?>%;"></div>
</div>
<div id="text">
      <div id="maxTemp">1200</div>
      <h1>Pressure</h1>
      <h1>Current: <?php echo $pressure; ?></h1>
      <div id="minTemp">0</div>
      <div id="backArrow"><a href="index.php"><img src="back.png" /></a></div>
</div>


</div>
</body>
</html>
```

Appendix K - humidity.php - Third sensor page on mobile site.

```
<html>
<head>
<link href="mobile.css" rel="stylesheet" type="text/css"/>
<meta name="viewport" content="width=320" />
<body>
<div id="header">
<h1>HOBO Weather</h1>
</div>
<div>

<?php
$dbhost = 'localhost';
$dbuser = 'nmlv2';
$dbpass = '';

$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error connecting to
mysql');

$dbname = 'hobo';
mysql_select_db($dbname);

$query = "SELECT * FROM readouts ORDER BY dateTime DESC LIMIT 1";

$result = mysql_query($query);
$num=mysql_numrows($result);

$dateTime = "";
if($result) {
 while($row = mysql_fetch_assoc($result)) {
       $dateTime=$row["dateTime"];
       $pressure=$row["pressure"];
       $temperature=$row["temperature"];
       $rh=$row["rh"];
       }
}
$current = $rh;
$goal = 100;
$mercury = ($current/$goal)*100;
?>
<div id="thermometer">
      <div id="humid" style="height:<?php echo $mercury; ?>%;"></div>
</div>
<div id="text">
      <div id="maxTemp">100 %</div>
      <h1>Humidity</h1>
      <h1>Current: <?php echo $rh; ?> %</h1>
      <div id="minTemp">0 %</div>
      <div id="backArrow"><a href="index.php"><img src="back.png" /></a></div>
</div>


</div>
</body>
</html>
```

Appendix L - about.php - Displays author information on mobile site.

```
<html>
<head>

<title>HOBO Mobile View</title>
<link href="mobile.css" rel="stylesheet" type="text/css"/>
<meta name="viewport" content="width=320" />

<body>
<div id="header">
<h1>HOBO Weather</h1>
</div>

<div>
     <h3>Mobile Site Author: Evan McKeon</h3>
     <h3>Full              Feature              Site:              <a
href="http://satnet.fgcu.edu/~nmlavand/projects/hobo/index.php">click
here!</a></h3>
     <div   id="backArrow"><a   href="index.php"><img   src="back.png"
/></a></div>
</div>

</body>
</html>
```